A probablistic analysis of the Game of the Goose

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

A probabilistic analysis of the Game of the Goose

Jan Friso Groote and Hans Zantema

14/04

ISSN 0926-4515

Computer Science Reports 14-04
Eindhoven, August 2014

A probabilistic analysis of the Game of the Goose

Jan Friso Groote and Hans Zantema

Department of Computer Science, Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
{j.f.groote,h.zantema}@tue.nl

August 22, 2014

Abstract

We analyse the traditional board game *the Game of the Goose*. We are particularly interested in the probability of the different players to win. We show that we can determine these probabilities for up to six players. Our original motivation to investigate this game came from progress in stochastic process theories which prompted us to ask ourselves whether those methods are capable of dealing with well known probabilistic games. As these games have large state spaces, this is not trivial. As a side effect we found that common wisdom about this game is not true.

1 Introduction

The Game of the Goose is a traditional board game. It comes in many variations and is commonly known, especially in Europe [1]. Essentially, it has a number of fields and the goal for each player is to be the first to reach the last field. Players move by throwing two dice and moving the indicated number of spots forward. On their way to the goal each player can encounter several difficulties, among which there are the well and the prison where they have to stay until released by another player.

In [5] the Game of the Goose is described as 'historically the most important spiral game ever devised' and it is claimed that the game stems from the Italy of Francesco de Medici (1574-1587), but similar games were already very popular in ancient history, especially among the Egyptians and the Greek.

Typical for all variants of the Game of the Goose is that there are 64 fields and two or more players. As the game is solely determined by throwing a pair of dice, no strategy is involved. Hence, the probability for each player to win the game is completely determined. An obvious question is whether one can determine these probabilities. That is the main issue of this paper.

Contrary to probabilistic games, the question to determine the winner in a strategy game has always attracted a lot of attention. And although it is not (yet) known which player has a winning strategy in the great games (chess, checkers, go), there are many strategy games for which this is known. Examples are four-in-a-row (or connect-four) [7] and restricted versions of awari [4].

So, we set out to determine the winning probabilities for each player of the 'Old Dutch' variant of the Game of the Goose (het oud-hollands ganzenbord), although even for this game there are different sets of rules and we simply made an arbitrary choice among those. One of the aspects that we ignore is the payment of a small fines ('fiches') that occur in some variants. The precise rules that we use are described in section 2. A typical layout of the playing board is shown in figure 1.

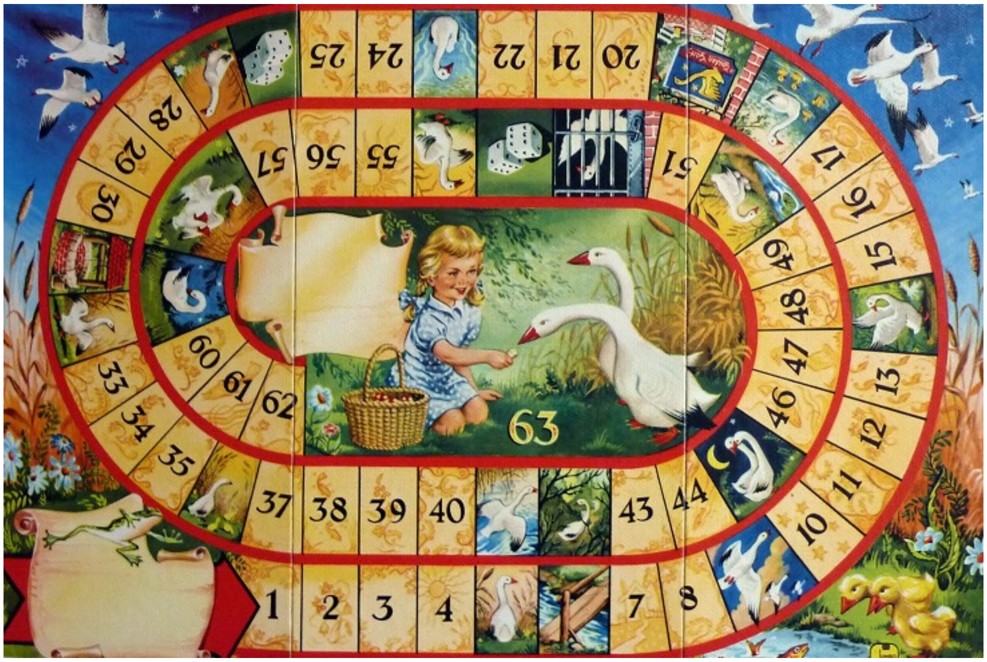

Figure 1: A Dutch version of the the Game of the Goose from appr. 1960

The most straightforward way to determine the winning probabilities is by simulating the game randomly a huge number of times and counting how often each player wins. We observed that convergence of winning probabilities is slow, and it is hard to determine the exact precision of the obtained probabilities. But we used this technique to verify the outcomes that we obtained in the way described below.

In this paper we calculate the probabilities by generating the complete state space of the game. Each legitimate way to put the players on the board together with the information who has the next turn constitutes a state. For each state and each player we introduce a variable expressing the probability to win the game for that player in that state. By formulating the relations between these probabilities, we get n linear equations among n variables, where n is the size of the state space. The number of states grows exponentially with the number of players. For the game with two players, this n is around 4000 and for five players there are 885 million.

Using Mathematica [8] we can solve the two player game exactly and the three player game using numeric approximation. The four player game could be solved using Matlab [3] with the induced dimension reduction (IDR) method as an extension package [6]. We managed to solve the game for five players with an ad-hoc fixed point algorithm. Establishing the winning probabilities for six or more players is out of our reach at the moment.

Inspecting these probabilities confirms many intuitive ideas about the game, but also lead to several surprising observations. The most surprising was that for the game with two players there is a substantial probability (23%) for the game to end in a draw, where one player is in the prison and the other is in the well. For all investigated numbers of players the first player has a small but definitive bias to win the game, which increases when more players join the game. For two players it is less than 2% and for five players it is almost 10%. It is also interesting to observe how the positions of the players on the board influence the winning probability. For two players we provide 3-D diagrams showing how these probabilities fluctuate depending on the relative position of the players on the board. From this one can for instance make the rather counter-intuitive observation that if one of the players ends in the well, this hardly increases the probability for the other player to win.

It is of course good fun to determine the winning probabilities for the Game of the Goose, but there is a more serious side to such an endeavour. Many real life phenomena can be described as probabilistic games. If we can analyze large games, we can analyze such real life phenomena as well. There are plenty of fully random games that can serve as useful playground. And if such games have been elaborated, there are still the games that combine strategy and randomness to be explored.

Acknowledgement. We thank Michiel Hochstenbach for his fruitful remarks, especially, for pointing out the IDR extension package for Matlab to solve large systems of linear equations.

2 The rules

Before presenting the results we have to determine the precise rules of the Game of the Goose. As it is a traditional game, it has been described in several sources. Although most ingredients of the game are the same in all sources, there are some minor differences. Here we fix a version that covers all interesting ingredients we met, and that is as close as possible to the various sources we considered.

- There are 64 positions, numbered from 0 to 63. All players start on position 0. The first player that reaches position 63 wins.
- A step of a player consists of throwing two 6-sided dice and by moving forward as many steps as there are spots shown.
- In case the resulting position is already occupied by another player, the player has to go back to its original position.
- A player only wins when he lands on position 63 exactly; if the number thrown is higher than required, the surplus is counted backwards from position 63.
- At positions 5, 9, 14, 18, 23, 27, 32, 36, 41, 45, 50, 54 and 59 there is a goose (see figure 1). If a player arrives at such a position, he will move the same amount of the roll again in the same direction. If this is again a position with a goose, this will be repeated.
- If from position 0 the dice show 3 and 6, the resulting position is 53. If from position 0 the dice show 4 and 5, the resulting position is 26. Note that if this rule would not exist, the player would jump immediately to the finish via the positions marked with a goose, when throwing 9 spots in the initial position.
- Position 6 is the bridge: a player arriving there will jump to 12.
- Position 19 is the inn: a player who is in the inn will stay there for one extra turn.
- Position 31 is the well: a player in the well will not play until another player arrives in the well.
- Position 42 is the maze: a player arriving there will go back to 30.
- Position 52 is the prison: just as with the well, a player in prison has to postpone participating in the game until another player ends up in prison releasing the first player.

- Position 58 is the death: a player arriving there has to start anew by going back to 0.

These rules are applied repeatedly. For instance, if a player is at position 46 and throws 4, he will move to 50. This is a goose, so he moves to 54. This is again a goose, so he moves to 58. This is death, so he has to start over by going back to 0. This combination of moves is considered as one move. If this position 0 is occupied since the turn before the other player came on death, this player will stay at 46. As another example, consider a player at position 60 throwing double 6, yielding 12 in total. By counting back he arrives at 54. Since this is a goose, he has to count back 12 more positions, arriving at 42. Since this is the maze, he goes to 30. Note that there is no upper bound on the number of allowed moves, but the probability that the game goes on forever equals zero.

If there are two players there are three possible outcomes: either player can win by arriving at position 63, but the game can also end in a draw if one player is in the prison and the other is in the well, since then no player is allowed to move. Note that when there are more than two players, one of the players will win, and there is no possibility for a draw.

Although we carefully described the rules of the game, experience shows that text can always be interpreted in more than one way. Therefore, we provide a formal description of the game in figure 2 using the process algebraic language mCRL2 [2]. This process algebraic language describes in essence in which sequences actions that can happen. In this case, the actions are **win**(p), **rest**(p) and **throw**(p, t_1, t_2) where p is the player, and t_1 and t_2 are the number of spots on both dice .

Players are represented by positive numbers ($\mathbb{N}^+$). The numbers $1, \ldots, N$ represent the actual players where N is the number of players. As indicated above, we use the letter p to represent a player. The functions *next* and *previous* provide the next and previous player in a round robin fashion. The fields on the board are given as natural numbers ($\mathbb{N}$) ranging from 0 (initial state) to number 64. Field 63 is the winning field. The field with number 64 is used for the inn. A player that arrives in the inn moves to position 64. When the player is at position 64 he moves to position 19 during his next turn, which represents waiting for one turn in the inn. The position of the players on the board is represented by a function *position*:$\mathbb{N}^+ \to \mathbb{N}$ that gives for each player its position on the board. The function *initial_positions* indicates the initial position on the board for each player. It is defined in the **eqn**-section as *initial_positions*$(p) = 0$, i.e., each player starts at position 0.

The process *PLAY* indicates the sequence in which the actions take place. It has two arguments. The first argument indicates the player that plays next, and the second argument is a function that for each player gives its position on the board. The behaviour of *PLAY* is given at its right hand side by if-then-else rules, denoted as $b \to x \diamond y$ indicating that if condition b holds, process x must be executed, and otherwise y is executed.

The first line, starting with the condition *position*(*previous*(p))$\approx$63 says that if the position of the previous player is 63, this previous player won the game, indicated by the action **win**(*previous*(p)) and after that nothing is done, indicated by the deadlock δ.

If this first condition does not hold, the second line applies, which is starting with the condition *position*$(p) \in \{31, 52\} \wedge \ldots$. It says that if the position of the current player p is 31 (the well) or 52 (the prison) and there is no other player in the well or prison ($\neg$*occ_twice*(p, *position*)) then the player must wait one turn, by carrying out the action **rest**(p) and continuing to play the game giving the turn to the next player and leaving the positions of all players unchanged (*PLAY*(*next*(p), *position*)).

If the second condition also does not hold, the third condition *position*$(p)\approx$64 can apply. This says that the player p is waiting in the inn. He automatically moves to position 19. This is denoted using the function update construction. The expression *position*$[p \to 19]$ represents the function *position*, except that it maps the argument p to 19.

```
eqn   $N = 4;$
      $\mathit{next}(p) = \mathit{if}(p{\approx}N, 1, p{+}1);$
      $\mathit{previous}(p) = \mathit{if}(p{\approx}1, N, p{-}1);$
      $\mathit{initial\_positions}(p) = 0;$
      $\mathit{adapt\_after\_63}(n) = \mathit{if}(n{>}63, 126{-}n, n);$

      $\mathit{next\_position}(p, \mathit{position}, t_1, t_2) =$
          $\mathit{if}(\mathit{position}(p){\approx}0 \wedge (t_1{\approx}4{\wedge}t_2{\approx}5 \vee t_1{\approx}5{\wedge}t_2{\approx}4), \mathit{if}(\mathit{occ\_twice}(p, \mathit{position}[p{\to}53]), 0, 53),$
          $\mathit{if}(\mathit{position}(p){\approx}0 \wedge (t_1{\approx}3{\wedge}t_2{\approx}6 \vee t_1{\approx}6{\wedge}t_2{\approx}3), \mathit{if}(\mathit{occ\_twice}(p, \mathit{position}[p{\to}26]), 0, 26),$
              $\mathit{next\_position}_2(p, \mathit{position}[p{\to}\mathit{adapt\_after\_63}(\mathit{position}(p){+}t_1{+}t_2)],$
                  $\mathit{if}(\mathit{position}(p){+}t_1{+}t_2{>}63, -t_1{-}t_2, t_1{+}t_2), \mathit{position}(p))));$

      $\mathit{next\_position}_2(p, \mathit{position}, \mathit{throw}, \mathit{old\_position}) =$
          $\mathit{if}(\mathit{position}(p){\in}\{5, 9, 14, 18, 23, 27, 32, 36, 41, 45, 50, 54, 59\},$
              $\mathit{next\_position}_2(p, \mathit{position}[p{\to}\mathit{adapt\_after\_63}(\mathit{position}(p){+}t)],$
                  $\mathit{if}(\mathit{position}(p){+}t{>}63, -t, t), \mathit{old\_position}),$
          $\mathit{if}(\mathit{position}(p){\approx}6, \mathit{next\_position}_2(p, \mathit{position}[p{\to}12], t, \mathit{old\_position}),$
          $\mathit{if}(\mathit{position}(p){\approx}19, \mathit{next\_position}_2(p, \mathit{position}[p{\to}64], t, \mathit{old\_position}),$
          $\mathit{if}(\mathit{position}(p){\approx}42, \mathit{next\_position}_2(p, \mathit{position}[p{\to}30], t, \mathit{old\_position}),$
          $\mathit{if}(\mathit{position}(p){\approx}58, \mathit{next\_position}_2(p, \mathit{position}[p{\to}0], t, \mathit{old\_position}),$
          $\mathit{if}(\mathit{position}(p){\notin}\{31, 52\}{\wedge}\mathit{occ\_twice}(p, \mathit{position}), \mathit{old\_position}, \mathit{position}(p)))))));$

      $\mathit{occ\_twice}(p, \mathit{position}) = \mathit{occ\_twice\_rec}(p, \mathit{position}, 1);$
      $\mathit{occ\_twice\_rec}(p, \mathit{position}, \mathit{other}) = \mathit{if}(\mathit{other}{<}N, \mathit{occ\_twice\_rec}(p, \mathit{position}, \mathit{other}{+}1), \mathit{false}){\vee}$
                        $(\mathit{other}{\not\approx}p \wedge$
                         $(\mathit{position}(p){\approx}\mathit{position}(\mathit{other}) \vee$
                          $\mathit{position}(\mathit{other}){\approx}19{\wedge}\mathit{position}(p){\approx}64 \vee$
                          $\mathit{position}(\mathit{other}){\approx}64{\wedge}\mathit{position}(p){\approx}19));$

proc  $\mathit{PLAY}(p{:}\mathbb{N}^+, \mathit{position}{:}\mathbb{N}^+{\to}\mathbb{N}) =$
          $(\mathit{position}(\mathit{previous}(p)){\approx}63){\to}\mathbf{win}(\mathit{previous}(p)){\cdot}\delta{\diamond}$
          $(\mathit{position}(p){\in}\{31, 52\} \wedge \neg\mathit{occ\_twice}(p, \mathit{position}))$
                                                  ${\to}\mathbf{rest}(p){\cdot}\mathit{PLAY}(\mathit{next}(p), \mathit{position}){\diamond}$
          $(\mathit{position}(p){\approx}64){\to}\mathbf{rest}(p){\cdot}\mathit{PLAY}(\mathit{next}(p), \mathit{position}[p{\to}19]){\diamond}$
          $\sum_{t_1, t_2{:}\mathbb{N}^+}.(t_1{\leq}6 \wedge t_2{\leq}6){\to}\mathbf{throw}(p, t_1, t_2){\cdot}$
                      $\mathit{PLAY}(\mathit{next}(p), \mathit{position}[p{\to}\mathit{next\_position}(p, \mathit{position}, t_1, t_2)]);$

init  $\mathit{PLAY}(1, \mathit{initial\_positions});$
```

Figure 2: An MCRL2 description of the Game of the Goose

If none of the three cases above apply, the player p throws two dice. This is represented using the sum operator $\sum_{t_1,t_2:\mathbb{N}^+}(t_1{\leq}6{\wedge}t_2{\leq}6){\rightarrow}\ldots$ which says that positive values for t_1 and t_2 are selected that satisfy the condition. Then the action **throw**(p, t_1, t_2) happens representing that player p throws a die with number t_1 and one with number t_2. Subsequently, the game continues where the next player gets a turn, and where the position of the current player is updated using the rather complex function *next_position*$(p, position, t_1, t_2)$. This function is defined in the **eqn**-section and it is explained below.

The function *next_position*$(p, position, t_1, t_2)$ calculates the next position of player p where he throws both t_1 and t_2 spots, given that the current position of the players is given by *position*. If p is at the initial position and a 5 and 4, or a 6 and 3 are thrown, player p moves to position 53, resp. 26 unless there is already a player occupying this field. In the latter case, the player stays at position 0. The expression *occ_twice*$(p,$*position*$[p{\rightarrow}53])$ is used to check that if player p moves to position 53, the position of player p is occupied twice. Note that in the definition of *occ_twice*, there is an extra check whether position 19 and 64 are both occupied. As both positions represent the inn, this also counts as a single field having a double occupancy.

If the special initial case described above does not apply in the definition of the function calculating the next position *next_position*$(p, position, t_1, t_2)$, its behaviour is defined by the auxiliary function *next_position*$_2(p,$*position*$,$ *throw*$,$ *old_position*$)$. It yields the ultimate position of player p on the board when player p did make an initial move (which is already reflected in *position*) where the dice showed the value *throw* (but this value is negative if p is moving backward) and *old_position* is the position where p came from. Note that the use of *next_position*$_2$ is tricky, because the player can have to move backwards when overshooting field 63.

In the definition of *next_position*$_2$ all remaining special rules of the game are dealt with. The first *if* deals with the 13 positions where the player can move the same number of moves ahead (or backwards). The second condition (*position*$(p){\approx}6$) deals with the situation where the player is at the bridge, and he must move to position 12. The third condition *position*$(p){\approx}19$ represents the player entering the inn. He is moved to the 'resting room' at position 64. The fourth condition *position*$(p){\approx}42$ indicates that the player is in the maze. He must move to position 30. The fifth condition *position*$(p){\approx}58$ corresponds to the situation where the player dies. He must restart by moving to position 0. The last condition applies when no subsequent move of the player is possible. It is checked whether the move of the player will lead to a double occupancy of fields (except for the well and the prison at positions 31 and 52 which can have more than one occupant. If there is a double occupancy the player moves to its old position, and otherwise it moves to the new position.

The mCRL2 tools allow to simulate the game and generate a full state space for the game which consists of all reachable configurations of players on the board. When interpreting the **throw** actions as being able to happen with probability $\frac{1}{36}$ the winning probabilities can be calculated by interpreting the state space as a discrete Markov chain. If a **win** or **rest** action can happen, no other actions are possible and therefore, one can consider these actions as happening with probability 1.

3 Analysis of a simple game

In this section we introduce an extremely simplified version of to Game of the Goose in order to illustrate how we obtain the probabilities. The rules of the game are simple. There are two players that start at position 0. The first player that reaches position 2 wins the game. Each player throws a two sided coin with no or one dot, and he will move zero or one positions forward in accordance with the value thrown. The game is illustrated in figure 3.

The game can be described in mCRL2 as follows:

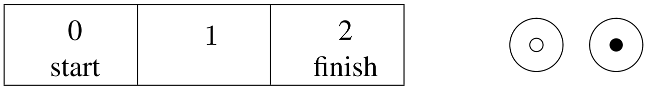

Figure 3: A simple two player game

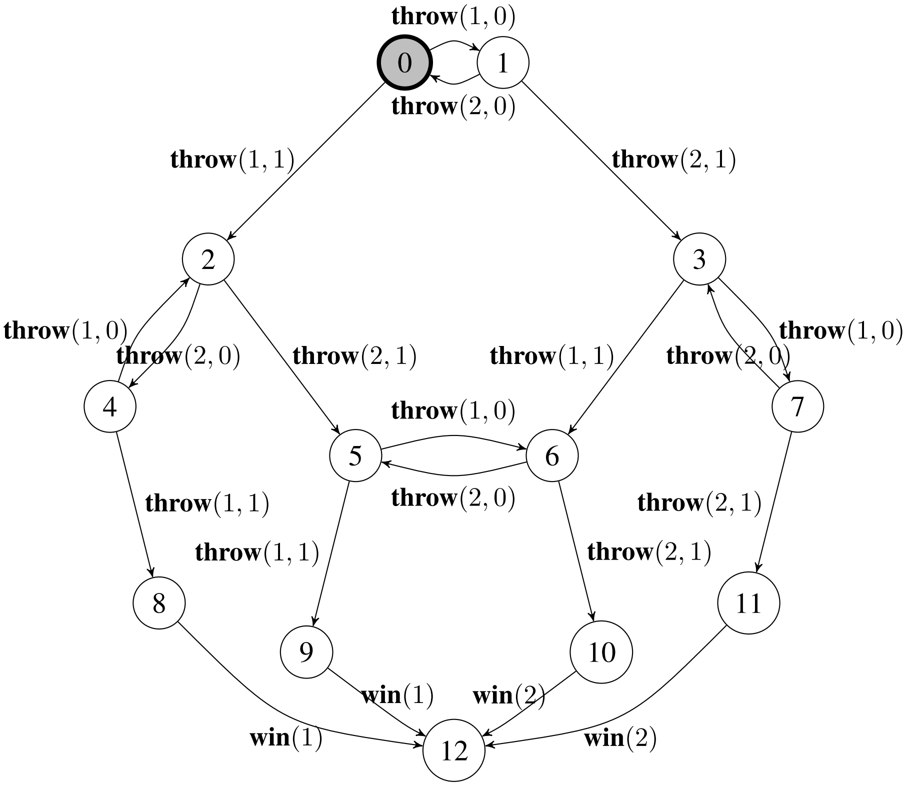

Figure 4: The state space of the simple game

```
proc    $PLAY(p_1, p_2{:}\mathbb{N}, turn{:}\mathbb{N}^+) =$
        $(p_1{\approx}2){\rightarrow}\textbf{win}(1){\cdot}\delta\diamond$
        $(p_2{\approx}2){\rightarrow}\textbf{win}(2){\cdot}\delta\diamond$
        $((turn{\approx}1){\rightarrow}(\sum_{t:\mathbb{N}}.(t{<}2){\rightarrow}\textbf{throw}(1,t){\cdot}PLAY(p_1{+}t, p_2, 2))$
                     $\diamond\ (\sum_{t:\mathbb{N}}.(t{<}2){\rightarrow}\textbf{throw}(2,t){\cdot}PLAY(p_1, p_2{+}t, 1)));$

init    $PLAY(0, 0, 1);$
```

The state space of the game is depicted in figure 4. The initial state is light grey and has number 0. In the initial state player one either throws zero (**throw**$(1, 0)$) or one (**throw**$(1, 1)$) which are represented by arrows leading to respectively state 1 and 2. In states 1 and 2 the second player can make a move. Figure 4 gives a nice overview how the game can proceed. At states 8 and 9 player 1 wins the game (**win**(1)) and in state 10 and 11 player 2 wins (**win**(2)). State 12 is a deadlock state where the game is finished, corresponding to δ in the mCRL2 description.

We are now interested in the probability for player 1 to win the game when he is in state i. We denote this probability by p_i. Clearly, $p_8 = p_9 = 1$, and $p_{10} = p_{11} = 0$. The probability p_{12} makes no sense because in state 12 the game is finished. For all other probabilities p_i we can derive a simple linear equation. The probability to win in state 1 is $\frac{1}{2}p_1 + \frac{1}{2}p_2$ because player one has 50% chance to end up in state 1 and 50% chance to end up in state 2. If we spell out all equations we get the following set of linear equalities.

$$\begin{array}{lll}
p_0 = \frac{1}{2}p_1 + \frac{1}{2}p_2 & p_4 = \frac{1}{2}p_2 + \frac{1}{2}p_8 & p_8 = 1 \\
p_1 = \frac{1}{2}p_0 + \frac{1}{2}p_3 & p_5 = \frac{1}{2}p_6 + \frac{1}{2}p_9 & p_9 = 1 \\
p_2 = \frac{1}{2}p_4 + \frac{1}{2}p_5 & p_6 = \frac{1}{2}p_5 + \frac{1}{2}p_{10} & p_{10} = 0 \\
p_3 = \frac{1}{2}p_6 + \frac{1}{2}p_7 & p_7 = \frac{1}{2}p_3 + \frac{1}{2}p_{11} & p_{11} = 0
\end{array}$$

This set of linear equations is small and therefore easily solved, leading to $p_0 = \frac{16}{27} \approx 0.59$. In order to find the probability that player two wins the game we can use the same set of equations, except that we must take $p_8 = 0$, $p_9 = 0$, $p_{10} = 1$ and $p_{10} = 1$. This leads to the expected result of $\frac{11}{27}$ as the winning probability for player 2. Obviously, the first player has a substantially higher probability of winning the game.

There are other ways of deriving these winning probabilities. A straightforward way is to simulate the game sufficiently often, which gives an approximation of probabilities, although for games with huge state spaces, these probabilities tend to converge slowly.

Another is to derive the linear equations directly from the game, without generating an explicit state space. We define the probabilities q_{ijk} to represent the probability that player 1 wins the game, provided player i ($i \in \{1, 2\}$) has the next turn, player 1 is at position j and player 2 is at position k ($j, k \leq 2$). The probability q_{100} is equal to $\frac{1}{2}q_{200} + \frac{1}{2}q_{210}$ because player one has equal probability to stay at position 0 or move to position 1, after which it is player two's turn. By carefully analysing all board positions of the game we can derive the following set of equations. Note that some probabilities are left out, as such probabilities cannot be reached, such as p_{12k}. These probabilities represent situations where player one can play and has won. Note that the obtained equalities are in this case exactly those obtained via the state space. For the Game of the Goose the number of equations that we obtained in both ways were slightly different.

$$\begin{array}{lll}
q_{100} = \frac{1}{2}q_{200} + \frac{1}{2}q_{210} & q_{200} = \frac{1}{2}q_{100} + \frac{1}{2}q_{101} & q_{110} = \frac{1}{2}q_{210} + \frac{1}{2}q_{220} \\
q_{112} = 0 & q_{210} = \frac{1}{2}q_{110} + \frac{1}{2}q_{111} & q_{220} = 1 \\
q_{101} = \frac{1}{2}q_{201} + \frac{1}{2}q_{211} & q_{111} = \frac{1}{2}q_{211} + \frac{1}{2}q_{221} & q_{201} = \frac{1}{2}q_{101} + \frac{1}{2}q_{102} \\
q_{211} = \frac{1}{2}q_{111} + \frac{1}{2}q_{112} & q_{221} = 1 & q_{102} = 0
\end{array}$$

We used all three ways to establish the winning probabilities. The reason for this is that it is very hard to not make a mistake in precisely modelling even simple games. By modelling it in three different ways we could compare the results and increased our confidence that our results are correct.

4 Computations for two players

If we analyse the Game of the Goose for two players, we obtain a set of 4048 or 4078 linear equations depending on which method is used for generation. We can derive the following results regarding the winning probabilities for both players.

$$\frac{\begin{array}{c}
354578138124950186817810489217182791317787798724802614410029328485608903369477431635018222558\\
399309450507103459074238228376405712500904558023752239339280938603803457573412946045548555428\\
192408470799948053129350811785597975235328183601609649319429499349866185498239751673351341769\\
582839004783122090660713501153416711577247932684590810429682087674204514516206428023232773142\\
970683832903046790844706695907372586643039096740098702406743747498676783882062602493264926223\\
838302824908562507570250392570595827698114873011234704695646697896143603966602499097872351154\\
669089452354191599916709746676851938051878405719576742617594702275857493698645246959330233303\\
091445902939904912863840744790654661105175111195454449663058266233638363673073343450482998878\\
224889581332206333579009718721098139784479036279112434678742992137019681147035075938046701550\\
918055159448125308748921320289899333197006467347143802920860473826874931173910161985261833871\\
891348234745481157765616175520794275559824385481401595657510498531296821490135004997149723846\\
312298317137414865283481215837836732118455291175984574489259882269507968728081889027165529711\\
660915404995631330437820160918345491253783805093745020459199567996953546847609820992902762908\\
750628689522634308453656275384033760133459102330761797545360967662642992417761488637785343096\\
179837783567510615016827169189382638260325730601238974784021870275751361465251485833641380988\\
601239137220474152073594938576999714048510670106244854187988128408120363500526256376389738531\\
527536713654289142088651349218285889196358082646355336459464707565595257453153280748427467824\\
804125023919836067186411643984746979465748585196447014561870836245035624955049440432350263977\\
824403133461025845332976436704285089656500992657870174243661788423998434592130682667322269405\\
049490116752141370798765615956317551142249979497567046069585441301567890743987803069586801932\\
669973979851519762918559922723993392470947267582828358727134952210072937955951818226524755153\\
725945829745811214804332328703353196315111784798277410263357528661496628912933728773218555977\\
962282140759639089508990171920383431393394642405434973415711204469313507363576994397787067207\\
425052329110883835493651467371006199576012293462207442967865421344357444321357800987822349475\\
376878156929670045445507371744044527718465666904104107234441730939460632072987861228015467811\\
882360487748332166057469542836916432762990864234404289049048042068157866859460085236085608315\\
623668882716193909184091442443597083669348518632793885704717828545288254527572206829503888843\\
189830953755201841220373817976349249436975535253449029794712313365966457380552647526110163477\\
309530899123504784001358147399949405347287670149449824157437381968509825640543358543867976582\\
112932024845249858864855940315369112486827116397833485123121716079117281508469354925885409764\\
674924868096271749236366074963266182936247416433242594731524706999193767511101871330792678259\\
280382594971805430937807818516642167895701603935105753261288618047393431315053008943441431152\\
112314766320450756187900149495780206679762576019422616321340814720511095615601551532129272718\\
542739705072677966046511469321938635400662566613122323344475936596007879096566202575850204524\\
189676551943771629765498166351584803390675990234443292078393699164128871377784023956250048044\\
331552068284643624644046415414958009203064443229091249119093243635766772160882604554315879605\\
060831221307872086909952800271602369875618608782388082232510649386131941136637973880915801002\\
154572511895832800230367533198381720968690002760891453868273714176187898521616789993779960852\\
761931148923821002490024262343407522301166428721085616227115222756065531414838764999639026632\\
039441391801485622144284338102119223432872300849269879911935735471566830470688649086825183967\\
844807850946421646147394307730886861802798090936151774215069265819265138777116887920005066844\\
010830068931878504072308068647265799544023019246293329832623031453567259134862222156900665676\\
786448672835928798628020069958939032396484550567944379409480283
\end{array}}{\begin{array}{c}
900801560775964656126774493326889579738207808673590850978148447654100516265185014530331647892\\
963003549157974321305021253246292417449947288406287943081249483047759028012482686458088710035\\
667076237996138211920100304931630064917887932900820281965696801766239831673398209178450364993\\
664141772270799113017382685324511970805425293800448269497362017976312673299247677364939829441\\
356682373589121359929188239605113065014562579196471221602072740197329678855003920081111125071\\
562181004087297676021601547101904445991031881920268404465974551250191245335606564444955874790\\
579668842903908691859663445026042171718496802772890001540462085862285203596824508136312510930\\
848256782907319085401577950566480236492258368651403851498168474838648238887253361826278320974\\
211582989769501836744362197433090321629992797581960180227991741382084013179433356134386879465\\
593763466375933164532378142992032030676788266593780697459863590954425340611505393883463891170\\
971971064338366560257877714490386750914477743904362410695314284442222150192349474163351136665\\
216496309031156352596603819405940309695923665530872630461144194903750285125269104079009482059\\
449373800161294244270975727298452311247329735693384663705652207467764860758084365569131209291\\
229381384664939466470853167529986572849474795951706727760432071740948785034512160370763120637\\
339519772942758900998043467561936589952119274344302734511358915860188099372260220328866589088\\
156263594454570824879678437060954154474559388781342874179426124184682655203719929399964083174\\
163103613617183832005419973105753767668901642442011068968931764251547938499039025283281070521\\
626469361626355506807630054195079424417327991706187186940870807368776939235542166982376805364\\
554504077292848886835539066227774214224562642653978516302884675382241286557954761373473131260\\
398683754349784037651478869577494891197647440888866724744290474747707116650727026018201161485\\
468721477586155819906563951435363663409456413316958670299530968871213165410259673299779136449\\
546570382066086147033604221435389641473231140715280626878852774956863960299786109914398931518\\
303131212483531294493331159638610052553627850621091892922181967442078061555180719269959611691\\
375488038095571656477654073523484609997634483232600502379658428980870346360321738192432670388\\
030086044391330549464290582334683133818870134666528351268436864593343373928316778450239320151\\
490616017571218225599667338674345549599369213030835046237448574463407637323433037344642693651\\
701648988954227861867925754021764324627782273822196813395500498990947900775262466674566797502\\
538208692635181129126570140804676875824884483807381185441791415008023307041497487737249042310\\
505372990270640822151972277764575305052107974872293504455660045611058486209180538423771137278\\
665443687822206785250296004711649547019031985510776338368259491949746485804668295542587032964\\
330180313351672274528115527876257178151725307206824846802270199905183486827906401182641218870\\
282587165253232412527268173813719376856485718148441478527543006201364921512289327167966829752\\
840415669558102361080423578070292648605539487934572107269958992134149955478331466018851569593\\
718356612338981495161662932308349814571401207328017569896080483808315461551874907047767528969\\
200021893878774750118480746745401602759010790621652738798780702998291443204275453806422937174\\
486042713846526300307683386755779988859765442464426523581101010235589403184706181157183193989\\
204685507819243217476508781703035183919970452946772074659104050764081839159268568343117208670\\
821350611437447602059630356954342172567631055484692745718321278817807410839504476278638075658\\
957815114791859980138843196023327044410709069485441809674404861155840974443329069685443563451\\
000330064673984317988986377130965450283592011509535800247907476188834267070613796557088503934\\
717084645791126874032339497526018300237585065551096520626437434582945009801773336358068326941\\
285478875469093008017253189675909895843648482906710452459750563144869260744545121189657756675\\
3191653418733360283585073225627136513606612524945418565535662080
\end{array}}$$

Figure 5: The exact winning probability for player 1 in a 2 player game

Probability that player 1 wins the game	0.3936
Probability that player 2 wins the game	0.3799
Probability for a draw	0.2265

Note that there is a substantial 23% probability that one player will end up in prison and the other in the well, leading to a draw in the game.

For a two player game it is actually possible using Mathematica to obtain the exact solution of the set of equations. As a curiosity the precise winning probability is given in figure 5 as a quotient of two natural numbers, which is approximately

$$0.3936251373937573914028403448768445020070441350696.$$

This exact solution can be used to check whether exactly the same game has been modelled. Minor flaws in modelling the game, such as forgetting that a player must rest one turn in the inn, will not substantially influence the winning probabilities of the players, but it will also not lead to exactly the same solution as given in figure 5.

It is interesting to figure out how the winning probabilities evolve while a game is progressing. For a two player game this can be neatly visualised, see figure 6. Here three diagrams are depicted, all with a view from above and from the side. The upper diagram models the probability that player one will win the game, when it is his turn to make a move. In particular if player one is alone in the well or in the prison, he cannot move, and this situation is not part of this diagram. The second diagram depicts the probability of player one to win the game, when player two is about to move. The third diagram depicts the probability of ending up in a draw.

If player one moves forward, he moves to the back of the diagram. If player two moves forward, he moves to the right. There are only 47 positions in the diagram, because all fields where a player must continue to move forward (i.e., a goose, death, the bridge or the maze) have been removed.

Note that the upper two diagrams have a solid wall at the back. This corresponds to player one winning the game. Similarly, there is a valley at the right, corresponding with player two winning the game, which means that the probability of player one to win the game is 0. Observe that the closer player one is to the finish, the higher is his probability to win, and reversely, the closer player two is to the finish, the lower is the probability that player one will win. Remarkably, if player two is in the prison, there is a substantially higher probability for player one to win. But if player two is in the well, this hardly influences the probability for player one to win. The reason for this can be seen in the third diagram. If player two is in the well, there is a substantially increased probability that the game will end in a draw. The two spikes in the third diagram correspond to the situation where the game is actually in a draw.

There is much more detailed information in these diagrams. For instance that it is not advantageous to be very close to the finish. But we leave the detailed interpretation of these features to the reader.

5 Winning probabilities for more players

It is also possible to establish the winning probabilities when there are more than two players, but this is increasingly more difficult as the number of states is growing exponentially, approximately according to the formula Nc^N where $c{\approx}45$ and N the number of players. In table 1 the winning probabilities are provided. The winning probabilities marked with an 'x' can be calculated, but it is simply too time consuming to do so. The obtained number required a few months of continuous calculations.

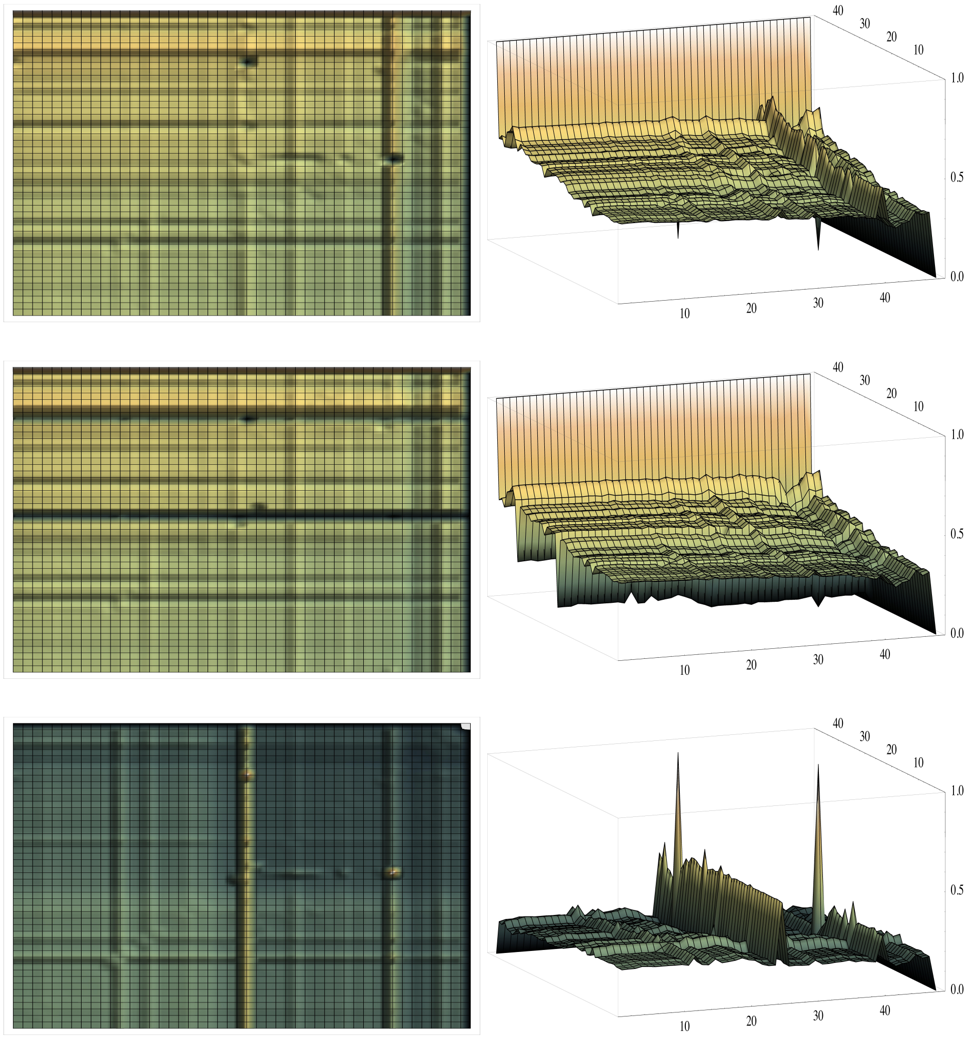

Figure 6: Visualisations of the winning probabilities of a two player game

We first solved the sets of linear equations using Mathematica [8]. This could be done exactly for two players and numerically for three players. For three and four players, using Matlab extended with the IDR package, we could solve the sets of obtained linear equations [3, 6]. For four players 400Gbyte of memory was required.

But we observed that the generated equations have a rather regular structure. For each variable p_i there is an equation of the shape

$$p_i = c_{i1}p_{i1} + \cdots + c_{ik}p_{ik}$$

where all c_{ij} are positive numbers smaller or equal than 1 and the solutions for all variables are in the interval $[0, 1]$. This linear set of equations can be viewed as a monotonic operator of which the solution can be obtained using fixed point iteration. Initially, all p_i are set to 1. Taking the equations a assignments, a new value for each p_i is repeatedly calculated until a fixed point is reached. This allowed to find the winning probability for player 1 in a five player game.

As it stands solving the game for six players is currently beyond our capabilities, although it is con-

	#equations	player 1	player 2	player 3	player 4
two player game	4078	0.39363	0.37999	-	-
three player game	$279\ 10^3$	0.34596	0.33290	0.32114	-
four player game	$16.4\ 10^6$	0.26695	0.25471	0.24408	0.23426
five player game	$885\ 10^6$	0.22039	x	x	x

Table 1: Winning probabilities when there are more than two players

ceivable that with a concerted effort, capable hardware and dedicated software this can be achieved. The number of required equations is estimated to be around $50\ 10^9$.

References

[1] H.C. Bolton. The game of goose. The Journal of American Folklore 8(29):145-150, 1985.

[2] J.F. Groote and M.R. Mousavi. Modeling and analysis of communicating systems. The MIT press. 2014.

[3] MATLAB version 7.10.0 (R2010a). The MathWorks Inc. Natick Massachussets, 2010.

[4] J.W. Romein and H.E. Bal. Solving the Game of Awari using Parallel Retrograde Analysis. IEEE Computer 38(10):26-33, 2003

[5] A.H. Seville. Tradition and Variation in the Game of Goose. Board Games in Academia III. (Proceedings of Colloquium in Florence), pp. 163-174, 1999.

[6] P. Sonneveld and M.B. van Gijzen. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems. SIAM Journal of Scientific Computing 31(2):1035-1062, 2008.

[7] J.W.H.M. Uiterwijk, H.J. van den Herik and L.V. Allis. A Knowledge-Based Approach to Connect- Four. The Game is Solved! Heuristic Programming in Artificial Intelligence: the first computer olympiad (eds. D.N.L. Levy and D.F. Beal), pp. 113-133. Ellis Horwood Limited, Chichester, 1989.

[8] Wolfram Research, Inc. Mathematica Version 8.0. Wolfram Research, Inc. Champaign, Illinois. 2010

Science Reports **Department of Mathematics and Computer Science**
Technische Universiteit Eindhoven

If you want to receive reports, send an email to: wsinsan@tue.nl (we cannot guarantee the availability of the requested reports).

In this series appeared (from 2012):

12/01	S. Cranen	Model checking the FlexRay startup phase
12/02	U. Khadim and P.J.L. Cuijpers	Appendix C / G of the paper: Repairing Time-Determinism in the Process Algebra for Hybrid Systems ACP
12/03	M.M.H.P. van den Heuvel, P.J.L. Cuijpers, J.J. Lukkien and N.W. Fisher	Revised budget allocations for fixed-priority-scheduled periodic resources
12/04	Ammar Osaiweran, Tom Fransen, Jan Friso Groote and Bart van Rijnsoever	Experience Report on Designing and Developing Control Components using Formal Methods
12/05	Sjoerd Cranen, Jeroen J.A. Keiren and Tim A.C. Willemse	A cure for stuttering parity games
12/06	A.P. van der Meer	CIF MSOS type system
12/07	Dirk Fahland and Robert Prüfer	Data and Abstraction for Scenario-Based Modeling with Petri Nets
12/08	Luc Engelen and Anton Wijs	Checking Property Preservation of Refining Transformations for Model-Driven Development
12/09	M.M.H.P. van den Heuvel, M. Behnam, R.J. Bril, J.J. Lukkien and T. Nolte	Opaque analysis for resource-sharing components in hierarchical real-time systems - extended version –
12/10	Milosh Stolikj, Pieter J. L. Cuijpers and Johan J. Lukkien	Efficient reprogramming of sensor networks using incremental updates and data compression
12/11	John Businge, Alexander Serebrenik and Mark van den Brand	Survival of Eclipse Third-party Plug-ins
12/12	Jeroen J.A. Keiren and Martijn D. Klabbers	Modelling and verifying IEEE Std 11073-20601 session setup using mCRL2
12/13	Ammar Osaiweran, Jan Friso Groote, Mathijs Schuts, Jozef Hooman and Bart van Rijnsoever	Evaluating the Effect of Formal Techniques in Industry
12/14	Ammar Osaiweran, Mathijs Schuts, and Jozef Hooman	Incorporating Formal Techniques into Industrial Practice
13/01	S. Cranen, M.W. Gazda, J.W. Wesselink and T.A.C. Willemse	Abstraction in Parameterised Boolean Equation Systems
13/02	Neda Noroozi, Mohammad Reza Mousavi and Tim A.C. Willemse	Decomposability in Formal Conformance Testing
13/03	D. Bera, K.M. van Hee and N. Sidorova	Discrete Timed Petri nets
13/04	A. Kota Gopalakrishna, T. Ozcelebi, A. Liotta and J.J. Lukkien	Relevance as a Metric for Evaluating Machine Learning Algorithms
13/05	T. Ozcelebi, A. Weffers-Albu and J.J. Lukkien	Proceedings of the 2012 Workshop on Ambient Intelligence Infrastructures (WAmIi)
13/06	Lotfi ben Othmane, Pelin Angin, Harold Weffers and Bharat Bhargava	Extending the Agile Development Process to Develop Acceptably Secure Software
13/07	R.H. Mak	Resource-aware Life Cycle Models for Service-oriented Applications managed by a Component Framework
13/08	Mark van den Brand and Jan Friso Groote	Software Engineering: Redundancy is Key
13/09	P.J.L. Cuijpers	Prefix Orders as a General Model of Dynamics

14/01	Jan Friso Groote, Remco van der Hofstad and Matthias Raffelsieper	On the Random Structure of Behavioural Transition Systems
14/02	Maurice H. ter Beek and Erik P. de Vink	Using mCRL2 for the analysis of software product lines
14/03	Frank Peeters, Ion Barosan, Tao Yue and Alexander Serebrenik	A Modeling Environment Supporting the Co-evolution of User Requirements and Design
14/04	Jan Friso Groote and Hans Zantema	A probabilistic analysis of the Game of the Goose