SIAM Review
Vol. 58, No. 1, pp. 143–155

A Probabilistic Analysis of the Game of the Goose[*]

Jan Friso Groote[†]
Freek Wiedijk[‡]
Hans Zantema[†‡]

Abstract. We analyze the traditional board game the *Game of the Goose*. We are particularly interested in the probabilities of the different players winning, and we show that we can determine these probabilities exactly for up to six players and using simulation for any number of players. Our original motivation to investigate this game came from progress in stochastic process theories, which prompted the question of whether such 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 find that some common wisdom about the game is not true.

Key words. probabilistic game, Game of the Goose, winning probabilities, recreational mathematics

AMS subject classifications. 97A20, 60C05

DOI. 10.1137/140983781

1. Introduction.

The Game of the Goose is a traditional board game that comes in many variations and is commonly known, especially in Europe [1]. In the game, the players are geese, which allows us to refer to them by the neutral gender (e.g., "its turn," "it wins"). The board has a number of fields and the goal for each goose is to be the first to reach the last field. On its turn, a player throws two dice and moves the indicated number of spots forward. On its way to the goal a player can encounter several difficulties, among which are the well and the prison, where the player is held until released by another player.

In [9] 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 popular in ancient history, especially among the Egyptians and the Greeks.

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 that each player wins the game is completely determined. An obvious question is whether one can determine these probabilities, and this is the main issue of this article.

Unlike probabilistic games, the question of determining the winner in a strategy game has always attracted a lot of attention. Although it is not (yet) resolved which

[*]Received by the editors August 25, 2014; accepted for publication (in revised form) April 1, 2015; published electronically February 4, 2016.
http://www.siam.org/journals/sirev/58-1/98378.html

[†]Department of Computer Science, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands (j.f.groote@tue.nl, h.zantema@tue.nl).
[‡]Faculty of Science, University of Nijmegen, 6500 GL Nijmegen, The Netherlands (freek@cs.ru.nl).

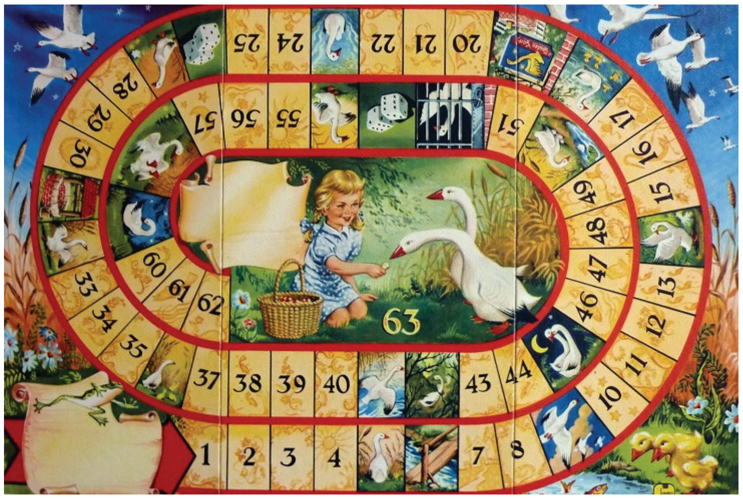

Fig. 1.1 *A Dutch version of the Game of the Goose from appr.* 1960. *(Copyright of the "Game of the Goose" image is owned by Koninklijke Jumbo B. V., Zaandam, The Netherlands, all rights reserved.)*

player has a winning strategy in the great games chess and go, there are many strategy games for which the winner is known, such as four-in-a-row (also called connect-four) [11] and restricted versions of awari [7]. With checkers the situation is mixed. American checkers, also called British draughts, played on an 8×8 board has been solved [8]. International checkers (10×10) and Canadian checkers (12×12) are still open problems.

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*). Unfortunately, even for this game there are different sets of rules and so we simply made an arbitrary choice among them. We ignore the payment of 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 1.1.

The most straightforward way to estimate the winning probabilities is by simulating the game randomly a huge number of times and counting how often each player wins. Doing this, we observed that the convergence of winning probabilities is slow: Simulating the game for 10^8 times typically gave precisions of only three to four digits. However, an advantage of such simulation is that the number of players is not a practical limit.

In this paper we also 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 about who has the next turn constitutes a state. For each state and each player we introduce a variable expressing the probability of winning the game for that player in that state. By formulating the relations between these probabilities, we obtain 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, n is around 4000, and for five players there are 885 million states.

Using *Mathematica* [4] we can solve the two-player game exactly as a fraction. For the three player game this is no longer possible, so instead we solve it using numeric approximation. The four player game can be solved using MATLAB [5] with the induced dimension reduction (IDR) method as an extension package [10]. 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 in this way is currently beyond our reach.

Inspecting these probabilities confirms many intuitive ideas about the game, but also leads to several surprising observations. The most surprising is that for the game with two players there is a substantial probability (23%) of the game ending in a draw, where one player is in the prison and the other is in the well. For all numbers of players investigated the first player has a small but definitive bias to win the game, which increases when more players join it. For two players, player one has a less than 4% higher probability of winning the game compared to the second player. For ten players, the first player has a 66% higher probability of winning the game than the tenth player.

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 positions of the players on the board. From this, one can, for instance, make the rather counterintuitive observation that if one of the players ends in the well, this only narrowly increases the probability of the other player winning.

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 endeavor. Rather different real life phenomena such as processor scheduling, weather forecasting, human interaction, and quantum mechanics can be viewed as probabilistic games. If we can analyze large games, we will have the techniques to analyze models of such real life phenomena as well. There are plenty of fully random games that can serve as a useful playground, and if all such games have been investigated, there are still the games that combine strategy and randomness to be explored.

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 by several sources. Although most ingredients of the game are the same in all sources, there are some differences. Here we fix a version that covers all the interesting ingredients we have encountered and that is as close as possible to the various sources 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 six-sided dice and moving forward as many steps as there are spots shown.
- If the resulting position is already occupied by another player, the player has to go back to its original position.
- A player only wins when it 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 1.1). If a player arrives at such a position, it will move the same count 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 did 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 position 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 does not take part in the game until another player arrives in the well, in which case the former player is free to move at the next turn and the latter player remains in the well until it is similarly released.
- Position 42 is the maze: a player arriving there will go back to position 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 death: a player arriving there has to start anew by going back to position 0.

These rules are applied repeatedly. For instance, if a player is at position 46 and throws 4, it will move to position 50. This is a goose, so it moves to position 54. This is again a goose, so it moves to position 58. This is death, so it has to start over by going back to position 0. This combination of throws is considered as one move. If position 0 is occupied since the turn before another player landed on death, this player will stay at position 46. As another example, consider a player at position 60 throwing double 6, yielding 12 in total. By counting back it arrives at position 54. Since this is a goose, it has to count back 12 more positions, arriving at position 42. Since this is the maze, it goes to position 30.

There is no upper bound on the number of moves and in principle the game can go on indefinitely. However, in the Game of Goose there is a positive probability that the game will end in a finite number of steps in every state. It can be shown that as there is a finite number of states, this implies that 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 can move. Note that when there are more than two players, one of the players will win and there is no possibility of a draw.

Although we have carefully described the rules of the game, experience shows that textual descriptions are virtually always incomplete and can be interpreted in multiple ways. A typical example with three players is the following. Player 1 is before the prison, player 2 is in the prison, and player 3 has passed the prison. On its turn, player 1 goes to prison, freeing player 2. Player 2 jumps to the position of player 3 and has to go back to prison. The question is whether player 1 or player 2 is now free. The listed rules above do not deal with this situation. The formal description below says that player 1 is free. A similar question occurs with the inn. If a player that leaves the inn is forced back into the inn, as it jumps to an occupied place, does it have to rest again one extra turn or not? (Answer: no.)

To make our rules absolutely clear, we provide a formal description of the game in Figure 2.1 using the process algebraic language mCRL2 [2]. This is a language that mathematically describes and analyzes the behavior of systems that send messages to each other. It is also quite suitable for describing games. Such languages stem from a tradition arising in [6], where it was observed that classical mathematics is not suitable for describing and studying digital communication among computers. The process algebra mCRL2 describes, in essence, in which sequences actions can happen. In this case, the actions are $\mathbf{win}(p)$, $\mathbf{rest}(p)$, and $\mathbf{throw}(p, t_1, t_2)$, where p is the player and t_1 and t_2 are the numbers of spots on each dice.

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

      $next\_position(p, position, t_1, t_2) =$
          $if(position(p) \approx 0 \wedge (t_1 \approx 4 \wedge t_2 \approx 5 \vee t_1 \approx 5 \wedge t_2 \approx 4), if(occ\_twice(p, position[p \to 26]), 0, 53),$
          $if(position(p) \approx 0 \wedge (t_1 \approx 3 \wedge t_2 \approx 6 \vee t_1 \approx 6 \wedge t_2 \approx 3), if(occ\_twice(p, position[p \to 53]), 0, 26),$
              $next\_position_2(p, position[p \to adapt\_after\_63(position(p)+t_1+t_2)],$
                  $if(position(p)+t_1+t_2 > 63, -t_1-t_2, t_1+t_2), position(p))));$

      $next\_position_2(p, position, throw, old\_position) =$
          $if(position(p) \in \{5, 9, 14, 18, 23, 27, 32, 36, 41, 45, 50, 54, 59\},$
              $next\_position_2(p, position[p \to adapt\_after\_63(position(p)+t)],$
                  $if(position(p)+t > 63, -t, t), old\_position),$
          $if(position(p) \approx 6, next\_position_2(p, position[p \to 12], t, old\_position),$
          $if(position(p) \approx 19, next\_position_2(p, position[p \to 64], t, old\_position),$
          $if(position(p) \approx 42, next\_position_2(p, position[p \to 30], t, old\_position),$
          $if(position(p) \approx 58, next\_position_2(p, position[p \to 0], t, old\_position),$
          $if(position(p) \notin \{31, 52\} \wedge occ\_twice(p, position), old\_position, position(p)))))));$

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

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

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

Fig. 2.1 *An MCRL2 description of the Game of the Goose.*

Players are represented by positive numbers ($\mathbb{N}^+$). The numbers $1, \ldots, N$ represent the actual players, where N is the total 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 players 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, it moves to position 19 during its next turn, which represents waiting for one turn in the inn. The position of the players is represented by a function $position{:}\mathbb{N}^+ \to \mathbb{N}$ that maps a player to its position on the board. The function *initial_positions* indicates the initial position on the board for each player and 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 behavior of $PLAY$ is given on its right-hand side by if-then-else rules, denoted as $b \rightarrow x \diamond y$, indicating that if condition b holds, process x must be executed, otherwise y is executed.

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

If this first condition does not hold, the second line applies, which starts with the condition $\mathit{position}(p) \in \{31, 52\} \wedge \cdots$. 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 ($\neg \mathit{occ\_twice}(p, \mathit{position})$), then the player must wait one turn by carrying out the action $\mathbf{rest}(p)$ and continuing to play the game giving the turn to the next player and leaving the positions of all players unchanged ($PLAY(\mathit{next}(p), \mathit{position})$).

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

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

The function $\mathit{next\_position}(p, \mathit{position}, t_1, t_2)$ calculates the next position of player p upon throwing $t_1 + t_2$ spots, given that the current position of the players is given by $\mathit{position}$. 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, respectively, 26, unless there is already a player occupying this field. In the latter case, the player stays at position 0. The expression $\mathit{occ\_twice}(p, \mathit{position}[p \rightarrow 53])$ is used to check whether if player p moves to position 53, the position of player p is already occupied. Note that in the definition of $\mathit{occ\_twice}$, there is an extra check for whether positions 19 and 64 are both occupied. As both positions represent the inn, this also counts as a single field having double occupancy.

If the special initial case described above does not apply in the definition of the function calculating the next position $\mathit{next\_position}(p, \mathit{position}, t_1, t_2)$, its behavior is defined by the auxiliary function $\mathit{next\_position}_2(p, \mathit{position}, \mathit{throw}, \mathit{old\_position})$. This yields the ultimate position of player p on the board when player p made an initial move (which is already reflected in $\mathit{position}$) where the dice showed the value throw (this value is negative if p is moving backwards), and $\mathit{old\_position}$ is the position from which p came. Note that the use of $\mathit{next\_position}_2$ is complicated, because the player must move backwards when overshooting field 63.

In the definition of $\mathit{next\_position}_2$, 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 ($\mathit{position}(p) \approx 6$) deals with the situation where the player is at the bridge and must move to position 12. The third condition $\mathit{position}(p) \approx 19$ represents the player entering the inn. It is moved to the "resting room" at position 64. The fourth condition $\mathit{position}(p) \approx 42$ indicates that the player is in the maze. It must move to position 30. The fifth condition $position(p) \approx 58$ corresponds to the situation where the player dies. It 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, otherwise it moves to the new position.

The mCRL2 tools allow us to simulate the game and generate for it a full state space 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 the Game of the Goose in order to illustrate how we obtain the probabilities. Two players 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 will move zero or one position forward in accordance with the value thrown. The game is illustrated in Figure 3.1.

0 start	1	2 finish

Fig. 3.1 *A simple two-player game.*

The game can be described in mCRL2 as follows:

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

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

The state space of the game is depicted in Figure 3.2. The initial state is light gray and has number 0. In the initial state player 1 either throws zero ($\mathbf{throw}(1,0)$) or one ($\mathbf{throw}(1,1)$), which are represented by arrows leading to, respectively, states 1 or 2. In states 1 and 2 the second player can make a move. Figure 3.2 shows how the game proceeds. At states 8 and 9, player 1 wins the game ($\mathbf{win}(1)$) and in states 10 and 11, player 2 wins ($\mathbf{win}(2)$). 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 of player 1 winning the game when it 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 of winning in state 0 is $\frac{1}{2}p_1 + \frac{1}{2}p_2$ because player 1 has 50% chance of ending up in state 1 and 50% chance of ending up in state 2. Writing out all equations leads to the

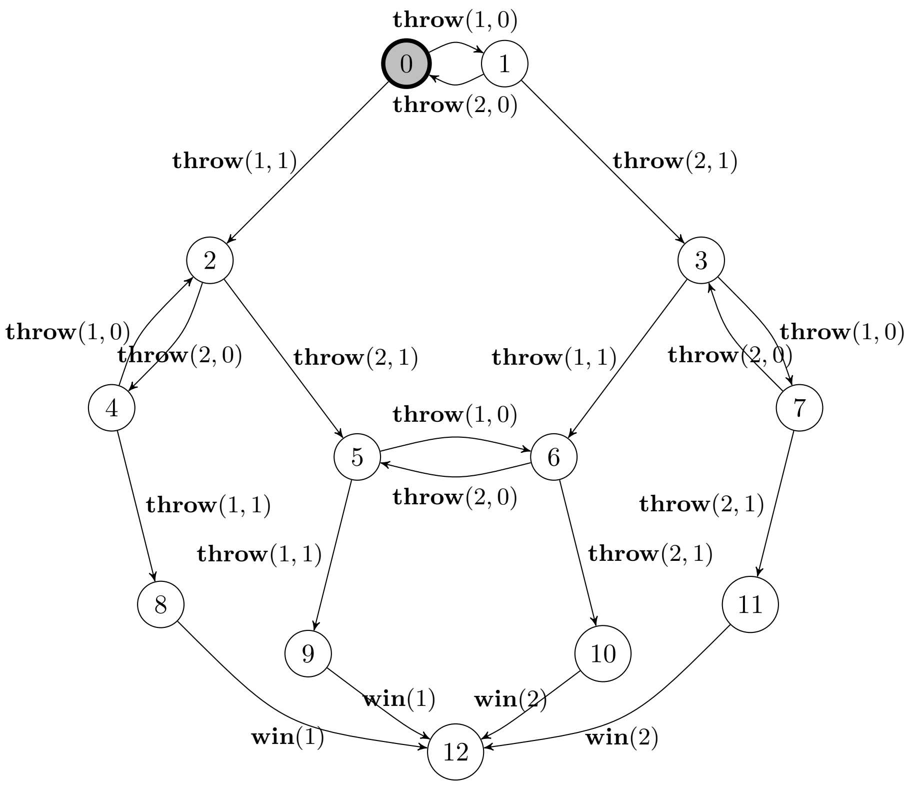

Fig. 3.2 *The state space of the simple game.*

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 2 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 that $\frac{11}{27}$ is the winning probability for player 2. As $\frac{16}{27} > \frac{11}{27}$, the first player has a higher probability of winning the game.

With a small change of the equations we can use them to calculate the average duration of the game. In every equation that corresponds to a turn of a player, 1 is added to the right-hand side. In every equation that represents winning or losing the game the right-hand side is set to 0. It follows that the average game duration is $\frac{16}{3} \approx 5.33$ moves.

We can also derive the linear equations directly from the game, without generating an explicit state space. We define the probability 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 1 has equal probability of staying at position 0 or moving to position 1, after which it is player 2's turn. By analyzing all board positions of the game one can derive the following set of equations. Note that probabilities for unreachable states, such as p_{12k}, are omitted. These probabilities represent situations where player 1 can play and has won. Note that the equalities obtained are in this case exactly those obtained via the state space:

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

An alternative approach is random simulation of the game where it is counted how often each player wins. The question asked is how often must one repeat the simulation to obtain a sufficient accuracy. Call the probability of winning the game for some player p, the number of simulation runs n, and the number of games that are won m. The observed probability of winning the game is then $\hat{p}=m/n$.

As the variance of a single coin flip with probability p is $p(1-p)$, the variance of simulating n games is $np(1-p)$. The standard deviation of the value of m over many runs of n simulations will be $\sqrt{np(1-p)}$. This means that the distribution of $\hat{p}$ is close to a normal distribution with mean value p and standard deviation $\sigma:=\sqrt{p(1-p)/n}$. For this reason, in 99% of the simulation runs the value of $\hat{p}$ will be in the interval $[p-2.58\sigma, p+2.58\sigma]$.

Now we can also use this in the other direction. If we find a certain value of $\hat{p}$, in 99% of the cases the true value of p will be in the interval $[\hat{p}-2.58\hat{\sigma}, \hat{p}+2.58\hat{\sigma}]$, where $\hat{\sigma}:=\sqrt{\hat{p}(1-\hat{p})/n}$. This means that $n>0.645\cdot 10^{2k}$ if we want to have k digits of precision, where we do not allow all digits to be correct in less than 1% of the cases.

We have used all three ways to establish the winning probabilities for more than two players and the first and last for more than two players. The reason for this is that it is easy to make a mistake in modeling even simple games. By modeling them in different ways we can compare the results, remove modeling mistakes, and increase our confidence that the final results are correct.

4. Computations for Two Players.

If we analyze the Game of the Goose for two players, we obtain a set of 4145 linear equations. We can derive the following results regarding the winning probability and average game duration for each player. The game duration is defined as the number of turns where the dice are thrown. If a player is in jail or in the well, or resting a second turn in the inn, this is not counted as a turn. All numbers are rounded in the ordinary way.

Probability that player 1 wins the game	0.3936
Probability that player 2 wins the game	0.3799
Probability of a draw	0.2265
Average game duration	29.0651

For a two-player game it is possible using *Mathematica* to obtain the exact solution of the set of equations. As a curiosity, the precise winning probability is given in Figure 4.1 as a quotient of two natural numbers, which is approximately

$$0.393625137393757391402840344876844502007044135069 6.$$

This exact solution can be used to check whether or not exactly the same game has been modeled. Minor flaws in modeling the game, such as forgetting that a player must

```
35457813812495018681781048921718279131778779872480261441002932848560890336947743163501822255 8
3993094505071034590742382283764057125009045580237522393392809386038034575734129460455485554 28
19240847079994805312935081178559797523532818360160964931942949934986618549823975167335134176 9
58283900478312209066071350115341671157724793268459081042968208767420451451620642802323277314 2
97068383290304679084470669590737258664303909674009870240674374749867678388206260249326492622 3
83830282490856250757025039257059582769811487301123470469564669789614360396660249909787235115 4
66908945235419159991670974667685193805187840571957674261759470227585749369864524695933023330 3
09144590293990491286384074479065466110517511119545444966305826623363836367307334345048299887 8
22488958133220633357900971872109813978447903627911243467874299213701968114703507593804670155 0
91805515944812530874892132028989933319700646734714380292086047382687493117391016198526183387 1
89134823474548115776561617552079427555982438548140159565751049853129682149013500499714972384 6
31229831713741486528348121583783673211845529117598457448925988226950796872808188902716552971 1
66091540499563133043782016091834549125378380509374502045919956799695354684760982099290276290 8
75062868952263430845365627538403376013345910233076179754536096766264299241776148863778534309 6
17983778356751061501682716918938263826032573060123897478402187027575136146525148583364138098 8
60123913722047415207359493857699971404851067010624485418798812840812036350052625637638973853 1
52753671365428914208865134921828588919635808264635533645946470756559525745315328074842746782 4
80412502391983606718641164398474697946574858519644701456187083624503562495504944043235026397 7
82440313346102584533297643670428508965650099265787017424366178842399843459213068266732226940 5
04949011675214137079876561595631755114224997949756704606958544130156789074398780306958680193 2
66997397985151976291855992272399339247094726758282835872713495221007293795595181822652475515 3
72594582974581121480433232870335319631511117847982774102633575286614966289129337287732185559 77
96228214075963908950899017192038343139339464240543497341571120446931350736357699439778706720 7
42505232911088383549365146737100619957601229346220744296786542134435744432135780098782234947 5
37687815692967004544550737174404452771846566690410410723444173093946063207298786122801546781 1
88236048774833216605746954283691643276299086423440428904904804206815786685946008523608560831 5
62366888271619390918409144244359708366934851863279388570471782854528825452757220682950388884 3
18983095375520184122037381797634924943697553525344902979471231336596645738055264752611016347 7
30953089912350478400135814739994940534728767014944982415743738196850982564054335854386797658 2
11293202484524985886485594031536911248682711639783348512312171607911728150846935492588540976 4
67492486809627174923636607496326618293624741643324259473152470699919376751110187133079267825 9
28038259497180543093780781851664216789570160393510575326128861804739343131505300894344143115 2
11231476632045075618790014949578020667976257601942261632134081472051109561560155153212927271 8
54273970507267796604651146932193863540066256661312232334447593659600787909656620257585020452 4
18967655194377162976549816635158480339067599023444329207839369916412887137778402395625004804 4
33155206828464362464404641541495800920306444322909124911909324363576677216088260455431587960 5
06083122130787208690995280027160236987561860878238808223251064938613194113663797388091580100 2
15457251189583280023036753319838172096869000276089145386827371417618789852161678999377996085 2
76193114892382100249002426234340752230116642872108561622711522275606553141483876499963902663 2
03944139180148562214428433810211922343287230084926987991193573547156683047068864908682518396 7
84480785094642164614739430773088686180279809093615177421506926581926513877711688792000506684 4
01083006893187850407230806864726579954402301924629332983262303145356725913486222215690066567 6
                    786448672835928798628020069958939032396484550567944379409480283
───────────────────────────────────────────────────────────────────────────────────────────────
90080156077596465612677449332688957973820780867359085097814844765410051626518501453033164789 2
96300354915797432130502125324629241744994728840628794308124948304775902801248268645808871003 5
66707623799613821192010030493163006491788793290082028196569680176623983167339820917845036499 3
66414177227079911301738268532451197080542529380044826949736201797631267329924767736493982944 1
35668237358912135992918823960511306501456257919647122160207274019732967885500392008111112507 1
56218100408729767602160154710190444599103188192026840446597455125019124533560656444495587479 0
57966884290390869185966344502604217171849680277289000154046208586228520359682450813631251093 0
84825678290731908540157795056648023649225836865140385149816847483864823888725336182627832097 4
21158298976950183674436219743309032162999279758196018022799174138208401317943335613438687946 5
59376346637593316453237814299203203067678826659378069745986359095442534061150539388346389117 0
97197106433836656025787771449038675091447774390436241069531428444222215019234947416335113666 5
21649630903115635259660381940594030969592366553087263046114419490375028512526910407900948205 9
44937380016129424427097572729845231124732973569338466370565220746776486075808436556913120929 1
22938138466493946647085316752998657284947479595170672776043207174094878503451216037076312063 7
33951977294275890099804346756193658995211927434430273451135891586018809937226022032886658908 8
15626359445457082487967843706095415447455938878134287417942612418468265520371992939996408317 4
16310361361718383200541997310575376766890164244201106896893176425154793849903902528328107052 1
62646936162635550680763005419507942441732799170618718694087080736877693923554216698237680536 4
55450407729284888683553906622777421422456264265397851630288467538224128655795476137347313126 0
39868375434978403765147886957749489119764744088886672474429047474770711665072702601820116148 5
46872147758615581990656395143536366340945641331695867029953096887121316541025967329977913644 9
54657038206608614703360422143538964147323114071528062687885277495686396029978610991439893151 8
30313121248353129449333115963861005255362785062109189292218196744207806155518071926995961169 1
37548803809557165647765407352348460999763448323260050237965842898087034636032173819243267038 8
03008604439133054946429058233468313381887013466652835126843686459334337392831677845023932015 1
49061601757121822559966733867434554959936921303083504623744857446340763732343303734464269365 1
70164898895422786186792575402176432462778227382219681339550049899094790077526246667456679750 2
53820869263518112912657014080467687582488448380738118544179141500802330704149748773724904231 0
50537299027064082215197227776457530505210797487229350445566004561105848620918053842377113727 8
66544368782220678525029600471164954701903198551077633836825949194974648580466829554258703296 4
33018031335167227452811552787625717815172530720682484680227019990518348682790640118264121887 0
28258716525323241252726817381371937685648571814844147852754300620136492151228932716796682975 2
84041566955810236108042357807029264860553948793457210726995899213414995547833146601885156959 3
71835661233898149516166293230834981457140120732801756989608048380831546155187490704776752896 9
20002189387877475011848074674540160275901079062165273879878070299829144320427545380642293717 4
48604271384652630030768338675577998885976544246442652358110101023558940318470618115718319398 9
20468550781924321747650878170303518391997045294677207465910405076408183915926856834311720867 0
82135061143744760205963035695434217256763105548469274571832127881780741083950447627863807565 8
95781511479185998013884319602332704441070906948544180967440486115584097444332906968544356345 1
00033006467398431798898637713096545028359201150953580024790747618883426707061379655708850393 4
71708464579112687403233949752601830023758506555109652062643743458294500980177333635806832694 1
28547887546909300801725318967590989584364848290671045245975056314486926074454512118965775667 5
                    31916534187333602835850732256271365136066125249454185655356620 80
```

Fig. 4.1 *The exact winning probability for player* 1 *in a two-player game.*

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 4.1.

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 visualized; see Figure 4.2. Here three diagrams are given, each with a view from above and from the side. The

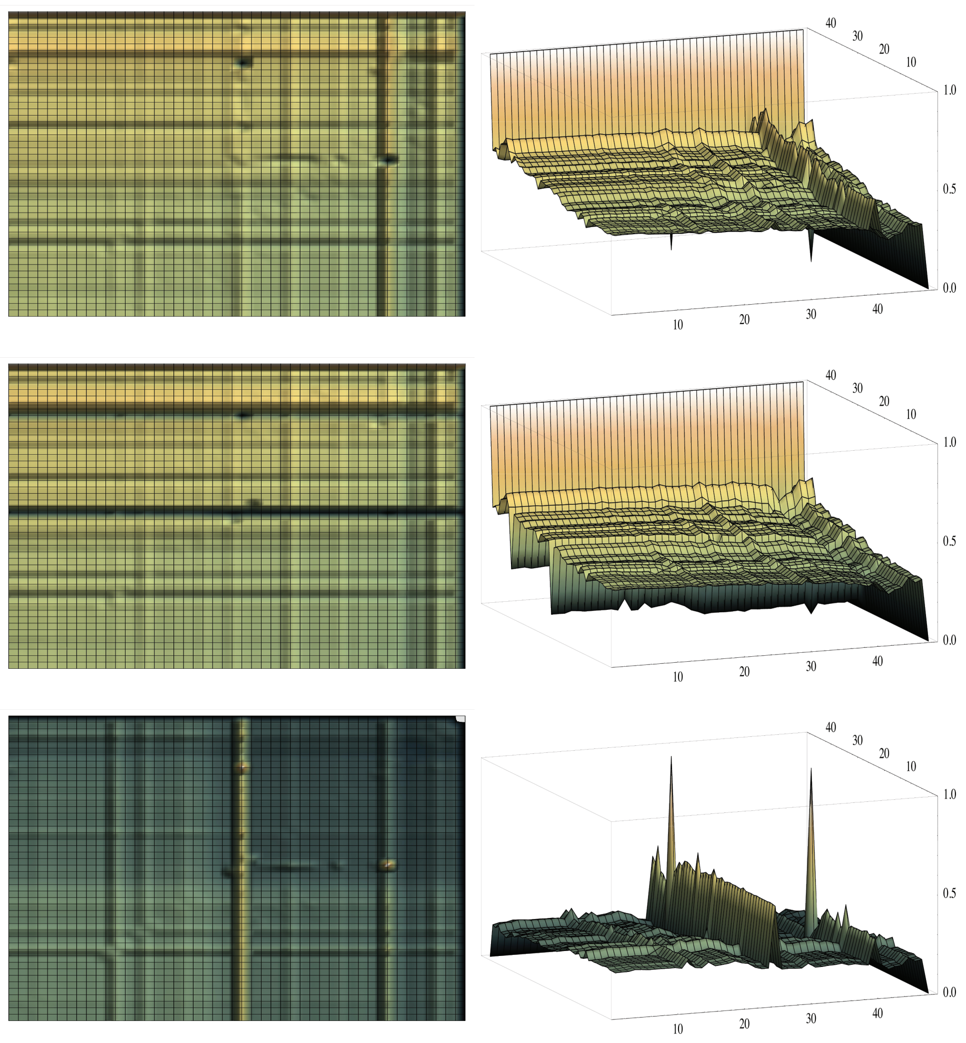

Fig. 4.2 *Visualizations of the winning probabilities of a two-player game.*

upper diagram models the probability that player 1 will win the game, when it is its turn to make a move. In particular, if player 1 is alone in the well or in the prison, it cannot move, and this situation is not part of this diagram. The second diagram depicts the probability of player 1 winning the game, when player 2 is about to move. The third diagram depicts the probability of ending up in a draw.

If player 1 moves forward, it moves to the back of the diagram. If player 2 moves forward, it 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 1 winning the game. Similarly, there is a valley at the right, corresponding to player 2 winning the game, which means that the probability of player 1 winning the game is 0. Observe that the closer player 1 is to the finish, the higher is its probability of winning, and conversely, the closer player 2 is to the finish, the lower is the probability that player 1 will win. Remarkably, if player 2 is in the prison, there is a substantially higher probability that player 1 will win, but if player 2 is in the well, this only narrowly influences the probability that player 1 will win. The reason for this can be seen in the third diagram. If player 2 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, it is not advantageous to be too close to the finish. 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 becomes increasingly more difficult as the number of states grows exponentially, approximately according to the formula Nc^N, where $c \approx 45$ and N is the number of players. In Table 5.1 the winning probabilities and the average game durations are provided. A game duration is defined as the number of moves where dice are thrown. Turns that are skipped are not counted. The numbers for more than five players were obtained by simulation only. To show that simulation is capable of solving games independently of the number of players, we have added the results for a ten player game as well, where we list the winning probabilities for all ten players.

Table 5.1 *Winning probabilities when there are two or more players.*

#players	#equations	Player 1	Player 2	Player 3	Player 4	Player 5	Average duration
2	4145	0.39363	0.37999	-	-	-	29.07
3	$2.79\ 10^5$	0.34596	0.33290	0.32114	-	-	39.42
4	$1.64\ 10^7$	0.26695	0.25471	0.24408	0.23426	-	44.10
5	$8.85\ 10^8$	0.22039	0.20875	0.19894	0.19012	0.18181	49.10
6	-	0.18986	0.17865	0.16944	0.16135	0.15390	54.35
10	-	0.12995	0.11992	0.11213	0.10568	0.10009	76.23
		0.09508	0.09049	0.08622	0.08218	0.07827	

We first solved the sets of linear equations using *Mathematica* [4]. This was done exactly for two players and numerically for three players. For three and four players, using MATLAB extended with the IDR package, we could also solve the sets of obtained linear equations [5, 10]. For four players 400GB of memory was required.

However, 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 than or equal to 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 for which the solution can be obtained using fixed point iteration. Initially, all p_i are set to 1. Taking the equations as assignments, a new value for each p_i is repeatedly calculated until a fixed point is reached. This allowed us to find the winning probabilities in a five player game.

The numbers in the table were obtained in different ways and on different computers. The numbers for five players were obtained by generating a state space with $8.85\ 10^8$ nodes and $3.11 \cdot 10^{10}$ transitions using the mCRL2 tools [2]. This took roughly

1.5 months and 300GB of memory on a Linux machine with Intel Xeon E5520 processors at 2.27GHz with 1TB of main memory. Solving the equations derived from the state space required approximately a week and 200GB of memory.

As it stands, solving the game using equations for six players is currently beyond our capabilities, although it is conceivable that with a concerted effort, capable hardware, and dedicated software this could be achieved. The number of required equations is estimated to be around $5.0 \cdot 10^{10}$.

The simulation results were obtained on a Linux machine with Intel E5-2670 processors at 2.6GHz. The simulation used ten processors simultaneously and required approximately a week to obtain the accuracy reported in this article.

Acknowledgments. 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. A shortened Dutch version of this article appeared in [3].

REFERENCES

[1] H. C. Bolton, *The Game of Goose*, J. Amer. Folklore, 8 (1985), pp. 145–150.

[2] J. F. Groote and M. R. Mousavi, *Modeling and Analysis of Communicating Systems*, The MIT Press, Boston, MA, 2014.

[3] J. F. Groote and H. Zantema, *De kans om Ganzenbord te winnen*, Nieuw archief voor de wiskunde, 5/15 (2014), pp. 234–239.

[4] *Mathematica, Version* 8.0, Wolfram Research, Inc., Champaign, IL, 2010.

[5] *MATLAB version* 7.10.0 (R2010a), The MathWorks Inc., Natick, MA, 2010.

[6] R. Milner, *A Calculus for Communicating Systems*, Lecture Notes in Comput. Sci. 92, Springer-Verlag, New York, 1980.

[7] J. W. Romein and H. E. Bal, *Solving the Game of Awari using parallel retrograde analysis*, IEEE Computer, 38 (2003), pp. 26–33.

[8] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen, *Checkers is solved*, Science, 317 (2007), pp. 1518–1522.

[9] A. H. Seville, *Tradition and variation in the Game of Goose*, in Board Games in Academia III (Proceedings of Colloquium in Florence), 1999, pp. 163–174.

[10] P. Sonneveld and M. B. van Gijzen, *IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations*, SIAM J. Sci. Comput., 31 (2008), pp. 1035–1062.

[11] 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!*, in Heuristic Programming in Artificial Intelligence: The First Computer Olympiad, D. N. L. Levy and D. F. Beal, eds., Ellis Horwood Limited, Chichester, UK, 1989, pp. 113–133.