Peeking Behind the Mask
Modeling Belief in a Game of *Mascarade*

Colin Shea-Blymyer [1]

Abstract

In this paper I study the beliefs of an agent playing a version of the game *Mascarade*. A player in this game must rely on memory and bluffs to convince others that one has the card one claims to have. The design of the game contains great mathematical structure, and represents an intuitively pleasing subclass of stochastic systems with partial observation. I introduce a formalization of the game, and present bounds on the game's action and state spaces. I then introduce an algorithm to track belief updates over actions and observations, and discuss alternatives to this algorithm in the context of computational and memory constraints. In demonstrating the bounds and beliefs associated with *Mascarade*, I illuminate an interesting set of stochastic processes, and draw connections between these processes and various fields of mathematics. Interpreted through a board game, these fields of theory can be more easily accessed by a broad population.

1. Introduction

Mascarade is a social bluffing card game for 2 to 13 players designed by Bruno Faidutti (Faidutti, 2014). In this game a player only knows which card they start with, and other players will try to decrease the confidence their opponents have in knowing which card lies in front of them. Thus, modeling a player's belief about which players have which cards is paramount in optimizing game play. The mechanics of the game are simple, but encode a rich structure that branches across probability, combinatorics, group theory, and graph theory.

[1]School of EECS, Oregon State University, Oregon, USA. Correspondence to: Colin Shea-Blymyer <sheablyc@oregonstate.edu>.

Proceedings of Oregon State University Probabilistic Graphical Models Course, Corvallis, Oregon, USA, 2020.

Figure 1. Contents of the game *Mascarade*. Character cards are pictured splayed across the top of the image. The goal of the game is to claim a total of 13 points represented by the coins seen here.

1.1. *Mascarade*

In *Mascarade* a player's objective is to use the powers of the card in front of them to be the first to obtain 13 coins. Gameplay starts with one card being randomly assigned to each player face-up, and each player starts with 6 coins. As play begins, all cards are turned face-down. Each card has an identity, and a power is associated with that identity. On each turn a player may take one of three actions:

1. *Announce* - claim that a certain card is in front of you and activate its power. The announcing player doesn't have to have the claimed card, but any number of other players (starting with the player to the announcing player's left, and continuing clockwise) may also claim to have that card and thereby challenge your claim. Once all challenges have been made, all players making a claim must reveal their card. Any player that revealed they do not have the claimed card loses one coin to a central pot. Any player that revealed they do have that card activates its power.

2. *Swap-or-not* - take another player's card along with your own. Place both cards under the table and shuffle them. Give one back to the other player and keep one for yourself. The shuffle need not be random, and you likely know which of the two cards you kept (though not the identity of those cards - you do not get to look at them).

3. *Look* - secretly look at the identity of your card.

The first few actions of the game must be swap-or-not to force players into states of partial knowledge. As the game progresses, the probability that a given player has a given card changes every time the card is swapped-or-not by another player, the card is revealed, and some times when another card is revealed - possibly resolving whether that card was involved in an earlier swap-or-not. To reach the game's objective, however, a player must claim to know which card they possess and announce their card. If all cards were equal we may assume that the best card to announce is that for which my probability of possessing it dominates that of other players. In this naive case, we can begin to see the value in statistically modeling the probabilities of players having certain cards.

Most cards, however, have unique powers. While most have a reward in coins as a component of their power, not all do. Other cards move coins in particular ways, or allow a player to gain knowledge, or obfuscate it. As such, assigning the rewards for actions can be very difficult. In this work, I focus on the properties of the belief process, and refrain from value modeling.

2. Related Work

The game of *Mascarade* can be straightforwardly understood as a discrete time stochastic process. The game's time is discretized on turns (or on players in the case of a challenged "announce" action), and the game is stochastic if the policies of other players is unknown. If the game is played with all cards revealed, an observer might model it as a Markov chain. Consequently, a player of such a game who possesses a notion of rewards for their actions can model the game as a Markov decision process (MDP) (Bellman, 1957). As per the rules, however, the cards remain concealed, and the true state of the game can only be partially observed through looking at one's own card, or as the result of a challenged claim. Appropriately, one might try to model the game as a partially observable Markov decision process (POMDP) (Åström, 1965). A POMDP augments an MDP with a probability distribution over possible states, and, if solved, provides optimal actions for each belief. While modeling *Mascarade* as a POMDP would provide a unique perspective on the game and a powerful platform for teaching decision modeling, the game's reward functions are complex and the POMDP belief update procedure is insufficient to capture the game's dynamics as is.

For a more expressive belief update, I consider the belief propagation algorithm for inference on graphical models (Pearl, 1982). In this scenario it provides a method for recalibrating the forward propagation of probability from previous states that, after observation, are known to have not occurred. While the belief propagation algorithm is not a perfect fit for belief updates in *Mascarade*, its core concepts and the fundamentals of its implementation on trees proved inspirational to this work.

3. Methodology

The formalization of *Mascarade* draws from a multitude of mathematical disciplines, including combinatorics, group theory, and probability. In this section, I formalize a simplification of the game as an observable system, and then extend that formalization to the partially observable case. In section 4.1 I use these formalizations to determine bounds on this game that will be useful for later analysis.

3.1. Fully Observable Formalization

A game is represented as the four-tuple $\Gamma =< \mathbf{J}, \mathbf{C}, \mathbf{S}, \mathbf{A} >$, where the set of players is $\mathbf{J} = \{J_0, \ldots, J_n\}$, the set of cards is $\mathbf{C} = \{C_0, \ldots, C_n\}$, the set of states is $\mathbf{S} = \{\sigma_0, \ldots, \sigma_m\}$, and the set of actions is $\mathbf{A} = \{\tau(i, j) \forall i \neq j\}$. Each turn in a game results in an action. In this fully observable case, the only actions are true swaps, as the results of announcing a card and looking at a card do not change the game's state.

3.2. Partially Observable Formalization

The partially observable case adopts the formalization from section 3.1, and introduces two core concepts. First, when an action τ is taken, the outcome is unknown, and will be modeled with a probability of 0.5 for occurring[1]. In a more complex model of the game, these transition probabilities would be determined by the active player's policy, according to that player's belief. Second, an observation $\lambda(i)$ can be made on a single item i in the current state. To illustrate, the observation $\lambda(1)$ can be interpreted as learning what card player J_1 has[2]. Thus $\lambda(1) \rightarrow 2$ signifies the observation that player J_1 is in possession of card C_2.

[1] In a Markov decision process, this probability would be associated with each action to form the state transition function.

[2] Since the assignment of cards to players is symmetric to the assignment of players to cards, it is valid to interpret $\lambda(1)$ as learning which player holds card C_1. However, this is inconsistent with the rules of *Mascarade*, so I adopt the former interpretation for the remainder of the text.

4. Results

In this section I introduce bounds derived from the structure of the game, discuss one implemented algorithm for belief updates, and propose two other such algorithms. Bounds on the algorithms will also be presented for later discussion.

4.1. Game Bounds

I use the structure of the game and the formalizations presented in section 3 to find bounds on the action and state space of a game. I also explore the structure inherent in the game.

4.1.1. Fully Observable Game Structure

Each state $\sigma \in \mathbf{S}$ is a unique bijection over players $\mathbf{J}$ and cards $\mathbf{C}$, and are therefore permutations on the bijection set. It follows that $\mathbf{S}$ is in the symmetric group S_n, giving $m = |\mathbf{S}| = n!$. Further, σ can be said to associate with the $n \times n$ permutation matrix M^σ whose entry $M^\sigma_{i,j} = 1$ if $\sigma(j) = i$, and 0 otherwise.

The fact that player J_0 possesses card C_3 is thus represented in all $M^\sigma : M^\sigma_{0,3} = 1$, corresponding with permutations that satisfy $\sigma(3) = 0$. In a four player game, this matrix must fit the pattern

$$M = \begin{bmatrix} 0 & 0 & 0 & 1 \\ * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \end{bmatrix}$$

A game Γ begins with the state σ_0 associated with the permutation matrix $M^0 = I$. An action $\tau(i, j)$ is a transposition on the state. A game's state σ after an action τ is given by the update $\sigma = \sigma\tau$. Each action can be said to move the state across an adjacent edge in the transposition graph G_n. This gives us $|A| = \binom{n}{2}$.

Though many of the properties of this game scale combinatorialy, the process itself is Markovian, so only the previous state needs to be remembered.

4.1.2. Partially Observable Game Structure

While the fully observable game may rely on the Markov property in order to reduce the order of information needed to store, the partially observable game does not satisfy the Markov property, and so must exploit the game's structure to gain traction. This and the inclusion of multiple players results in this game's deviation from an orthodox POMDP model. The partially observable game admits myriad representations — three of which form the foundations each belief update algorithm in section 4.2. I will address each of these in the following section.

4.2. Belief Update

In the partially observable game, the most difficult procedure in updating beliefs is the update as a result of an observation. In such cases, previous actions may be found to be impossible in retrospect, and balance of probability must then be recalculated. Fortunately, the structure of this game is a powerful tool for reducing the complexity of belief updates. In this section I introduce three methods for storing and updating beliefs. The first, which has been fully implemented, uses a tree to track all possible lineages of states. The second tracks only a distribution over possible states and the history of transitions, but increases the complexity of updates from observation. The third method employs a doubly stochastic matrix, which possesses desirable properties but does not lend itself to observation updates.

4.2.1. Probabilistic Branching Time Update

This update procedure, named after the alethic logic that inspired it, reflects the mechanics of Pearl's belief propagation algorithm in its observation updates. As shown in figures 2 and 3, the core data structure of this algorithm is an execution tree. Each level of the tree represents the outcome of a swap action. The root of the tree is the starting state, and each node contains a possible permutation matrix and its possibility. After a swap, each leaf node spawns one child node with the same state (representing the chance the swap did not occur) and another with the state formed by the product of the original state and the swap's associated transposition (representing the chance the swap did occur). Both child nodes inherit half of its parent's probability.

When an observation update occurs (depicted in figure 3) each leaf node is checked for compliance with the observation. Those whose states are rendered impossible by the update are removed from the tree, and their probability is considered to be 0. If both children are pruned from a parent node, then the parent node, too, is pruned, and so on, recursively, until no more nodes need to be removed from the tree. With respect to the belief propagation algorithm, this tree can be considered the factor graph, and the pruning is analogous to setting the potentials of each invalid node to 0, so outgoing messages from an invalid node's parent will not be diminished for its sibling. Next the tree is recalibrated with respect to its new structure, propagating its probabilities down until each leaf node receives an update.

As this algorithm stores the execution tree at each time step, the space required in on the order of 2^t, where t is the number of transpositions taken. Though observations do reduce the number of nodes tracked in the tree, an observation leaves $(n-1)!$ unique states feasible once all states are possible. Since observation updates check at least every leaf node, and propagate probabilities from the root, it too is

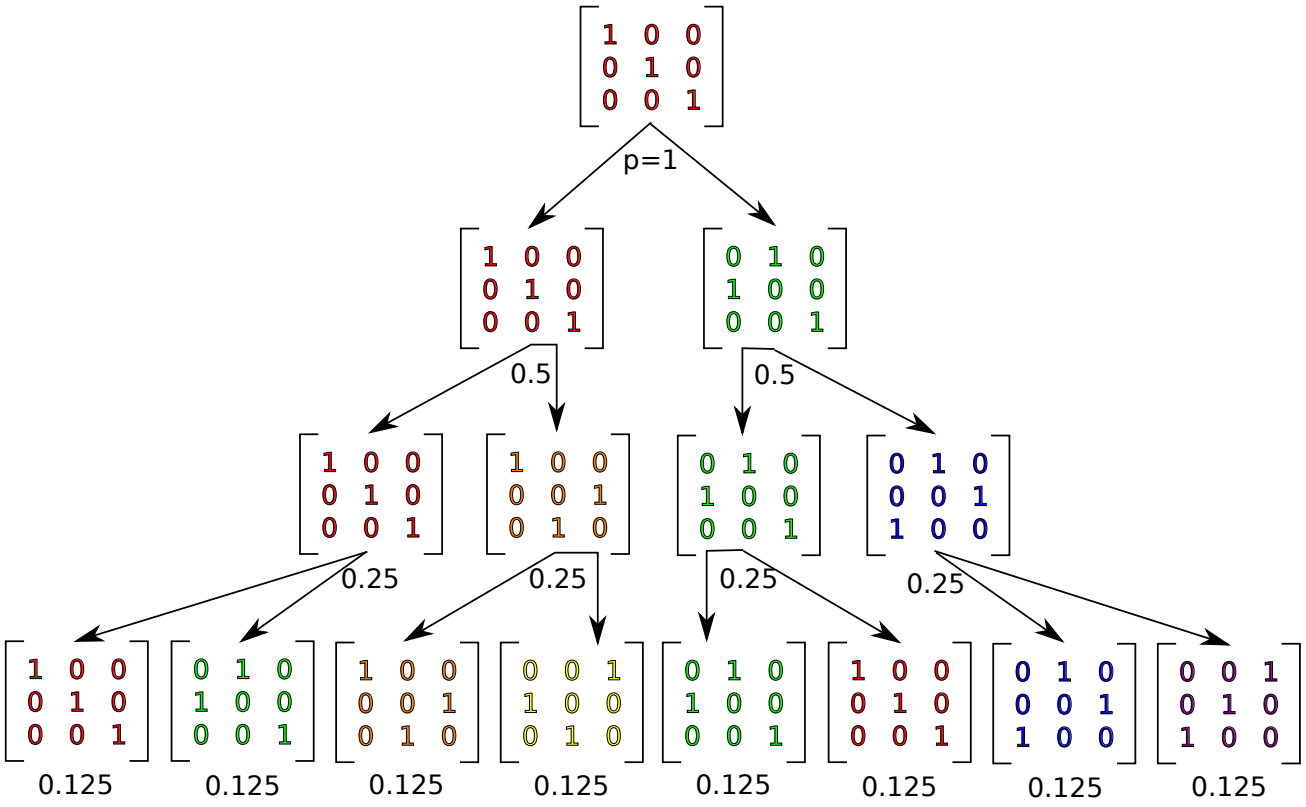

Figure 2. Illustration of the Probabilistic Branching Time Update after transitions for $n = 3$. The game begins with the known state (represented as the permutation matrix $M^0 = I_3$ with probability = 1) that each player starts with their originally assigned card. Next, player J_0's card is swapped with player J_1's card. With a transition probability of 0.5, the probability that the state remains the same is 0.5, as is the probability that the swap did, in fact, occur (resulting in the right-child permutation matrix). The probabilities propagate forward in this fashion after each swap is made. This game is generated by the permutation sequence $\sigma_0 * \tau(0,1) * \tau(1,2) * \tau(0,1)$.

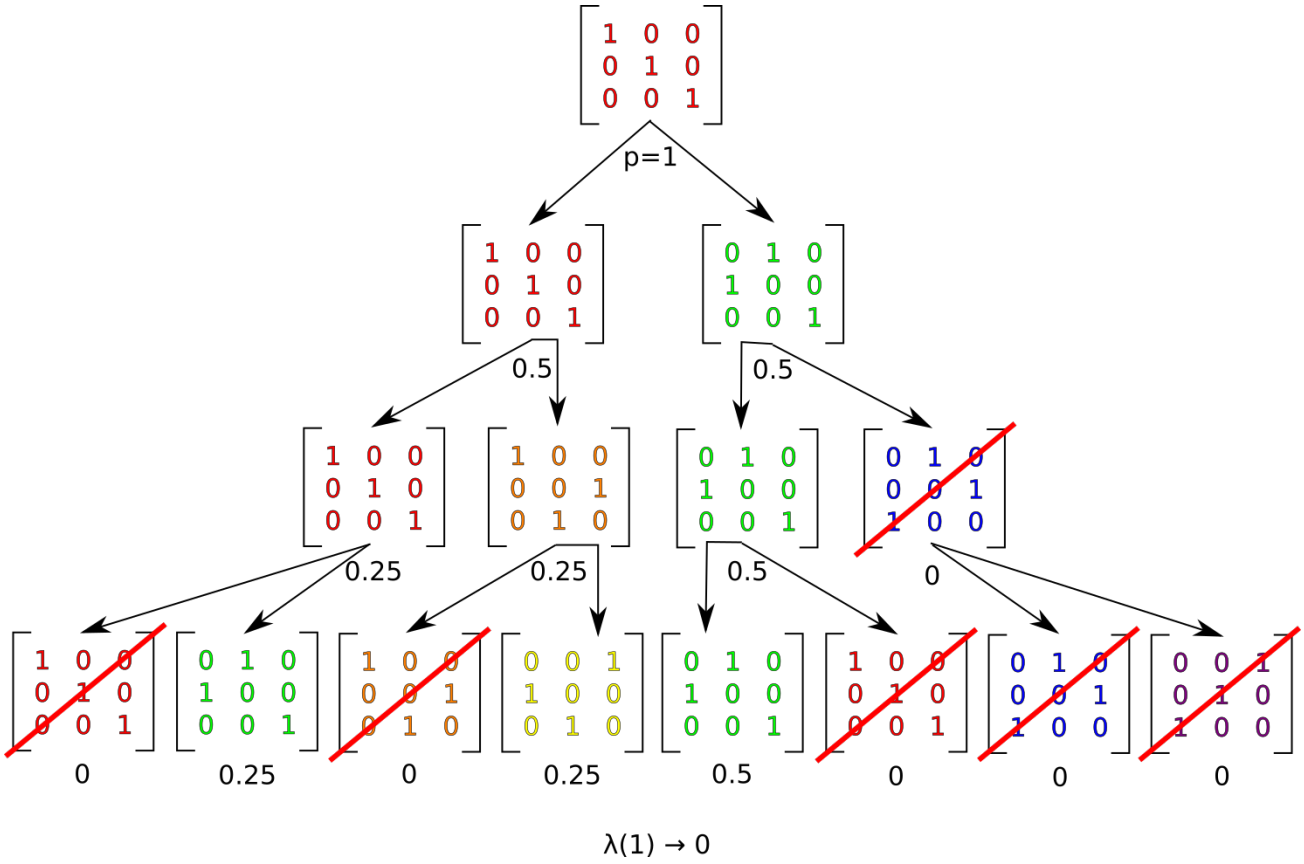

Figure 3. Illustration of the Probabilistic Branching Time Update after observation for $n = 3$. This game shows the one depicted in figure 2 after the observation $\lambda(1) \to 0$. This observation (i.e. that player J_1 has card C_0) eliminates the possibility of the multiple leaf states that don't match $M^{\sigma}_{1,0}$. Further, the rightmost state generated by the second transition is also found to be invalid. This implies that if the first swap did happen, then the second swap could not have. Once all invalid nodes are removed from the execution tree, probabilities can be propagated from the root node, resulting in an accurate update of the game's state. Future updates will only affect the remaining states and their new probabilities. This game is generated by the permutation sequence $\sigma_0 * \tau(0,1) * \tau(1,2) * \tau(0,1) * (\lambda(1) \to 0)$.

bounded by the order of 2^t.

This algorithm is intuitive, and well founded. An implementation of this algorithm can be found in this paper's supplementary materials.

4.2.2. Distribution Over States Update

An alternative to tracking all possible lineages of states through a game is to maintain a distribution over all possible states, and a history of transitions taken. This is similar to how POMDP models perform updates.

When a game begins, this algorithm must first produce the full list of states, and the transposition edges between them, i.e. the full transposition graph must be generated with edges mapped to swap actions and nodes mapped to states. Then the state corresponding with M^0 is assigned a probability of 1 and all other states are assigned a probability of 0. When a swap occurs, each state on the transposition graph splits its probability with the state at the other end of the edge associated with the current swap.

For observation updates the transposition graph must be iterated over time based on the recorded order of swaps. In this case the graphs may be considered directed acyclic probabilistic graphical models, and beliefs may be propagated after infeasible edges are pruned following a similar method described in the observation update of section 4.2.1.

This procedure can take a shortcut to the transposition graph by generating new states as they become possible (much like the tree in section 4.2.1), and merging non-unique nodes while associating the generating transposition that created the duplicates. Without this shortcut, this method faces an upper bound on the order of $tn!$ on storage. With the shortcut, this method has a storage upper bound of $\min(2^t, n!)$ for nodes and an additional $\frac{n!}{2}\binom{n}{2}$ in storage if the algorithm (eventually) caches all the edges in the transposition graph, or and additional $\min(2^t, n!)/2$ in complexity as it checks each newly generated leaf node for uniqueness.

4.2.3. Doubly Stochastic Matrix Update

While both previous methods track the likelihoods of states, the doubly stochastic matrix update represents each update as a change in the probability that player J_i has card C_j for each i and j. For instance, the doubly stochastic matrix representation of the game generated by the permutation sequence $\sigma_0 * \tau(0,1) * \tau(1,2) * \tau(0,1)$ (see figure 2) is

$$M = \begin{bmatrix} 0.375 & 0.375 & 0.25 \\ 0.375 & 0.375 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}$$

where $M_{0,0}$ represents the probability that player J_0 has card C_0, etc.

This representation has many desirable properties: it intuitively presents probabilities to a player in a manner that is actionable; transposition updates to the matrix reduces to simple matrix operations; and the matrix requires only $n \times n$ space. Further, by the Birkhoff-von Neumann theorem (Birkhoff, 1946), any doubly stochastic matrix can be represented by a convex combination of permutation matrices. In terms of probability, a doubly stochastic matrix can be thought of as the sum of all states scaled by their probability. This representation is known as the Birkhoff-von Neumann decomposition of the matrix.

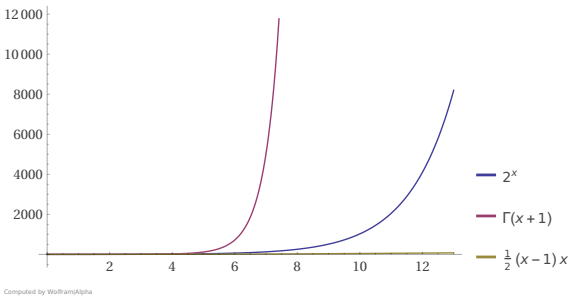

Figure 4. A comparison of some of the bounds found in this work.

As shown previously, knowledge of the current state is insufficient to produce knowledge of the state after an observation. With the Birkhoff-von Neumann decomposition in mind, one might consider retaining a history of doubly stochastic matrices and use the decomposition to produce the network structure introduced in section 4.2.2. Unfortunately, the decomposition is not unique, so one runs the risk of including a bogus history when building the network. To ameliorate this, one can check each decomposition against the previous, but this entails the same effort as rebuilding the tree from scratch.

5. Discussion

The doubly stochastic matrix update has very striking pros and cons. In its favor are the conveniences and interpretability of its form, the ease of transposition updates, and low storage costs. Against it lies the expense of performing an observation update. With these in mind it is clear that in a game where actions are evenly balanced between observations and transpositions, it is better to forgo this method. However, much can be gained with this method in a game where the number of transpositions is much greater than the number of observations.

Similar trade-offs exist for the distribution over states update and the probabilistic branching time update. The former performs better on average with a larger number of observations, but the maintenance and storage required on new

leaf nodes holds it back when many transpositions are being performed. The latter method may take more space, and longer games will make that space tend toward infinity, but it performs transposition updates quicker than the distribution over states method, and observation updates quicker than the doubly stochastic matrix method. In the average game, the optimal update is likely to be a hybrid between the unbounded tree in the early game, and the states distribution as the number of leaves in the tree surpasses the total number of states. To put this in perspective, in a game of 6 people (approximately half of the maximum number of players), the number of leaves on the tree would surpass the total number of states after 10 card swaps.

6. Conclusion

In this paper I have formalized the card game *Mascarade* as class of stochastic processes, derived bounds on the action and state space of of these processes, introduced and analyzed three algorithms for updating a belief state on these processes, and I've discussed cases where the introduced algorithms are more preferable than the others based on their performance. The bounds and algorithms rely on rich mathematical structure in this game, providing a unique and entertaining case study for the application of many veins of theory. I also implemented a belief update algorithm for the game, which can be used in future work to help model a player's decision policy.

Some small optimizations can be made to the algorithms implemented and proposed. When updating from a network, propagation need not start from the top of the tree, but only from one level deeper than the deepest pruned node. Also, as mentioned in the end of section 5, one can easily transition memory models from trees to networks when $2^t > n!$.

Finally, I have yet to believe that I've squeezed all the use out of the structure in this game. The patterns I've noticed in my experiments with the doubly stochastic matrix update method suggest there's some hidden key I've not yet discovered. It seems that the richer the structure, the more difficult it is to find exactly which structural features will be useful.

References

Bellman, R. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518.

Birkhoff, G. Three observations on linear algebra. *Univ. Nac. Tucumán. Revista A.*, 5:147–151, 1946.

Faidutti, B. Mascarade. [Board Game], 2014.

Pearl, J. *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science . . . , 1982.

Åström, K. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10 (1):174 – 205, 1965. ISSN 0022-247X. doi: https://doi.org/10.1016/0022-247X(65)90154-X. URL http://www.sciencedirect.com/science/article/pii/0022247X6590154X.