Fully-Nested Interactive POMDPs for Partially-Observable Turn-Based Games

Pol Rosello
Computer Science Department
Stanford University
Stanford, CA 94305
Email: prosello@stanford.edu

***Abstract*—Interactive POMDPs (I-POMDPs) are a useful framework for describing POMDPs that interact with other POMDPs. I-POMDPs are solved recursively in levels: a level-1 I-POMDP assumes the opponent acts randomly, and a level-k I-POMDP assumes the opponent is a level-(k-1) I-POMDP. In this paper, we introduce *fully-nested* I-POMDPs, which are uncertain about the physical state of the game, the level of their opponent, and the opponent's belief about both. This paper has three main contributions: it (1) introduces the framework for turn-based fully-nested I-POMDPs and shows how to reduce them to POMDPs; (2) motivates fully-nested I-POMDPs by introducing the game of *partially-observable nim* and solving it using SARSOP; and (3) shows empirically that increasing the level of a fully-nested I-POMDP does not become intractable for this game.**

I. Introduction

Nim is a mathematical game in which two players take alternate turns removing objects from distinct heaps. In one version of nim, there are two heaps on the board, each starting with the same number of objects. Each turn, a player selects one of the two heaps and removes as many objects from that heap as they wish, provided they remove at least one object. The goal of the game is to end one's turn leaving exactly one object on the board. Nim is a solved game: given any board state, there is a deterministic policy for one of the players which will guarantee that they win the game in the end [1].

In our work, we consider an alternative version of the game we call *partially-observable nim* (*PO-Nim*). In PO-Nim, the objective is the same as regular nim, but each heap is only visible to one of the two players. The challenge is to win the game by inferring how many objects are left in the opponent's heap. If a player attempts to remove more objects from the hidden heap than there exist, they lose their turn and incur a penalty. We formalize PO-Nim below:

- The physical *state* of the game at any point in time can be represented as $(n_{\text{own}}, n_{\text{opp}})$: the number of objects n_{own} left in the visible heap, and the number of objects n_{opp} left in the hidden heap. Players may be uncertain about the state of the game because they cannot directly observe the opponent's heap. The game ends when the state reaches $(1, 0)$ or $(0, 1)$. The initial physical state of the game is (N, N) for some N.
- Each round, the player takes an *action* (h, i) consisting of the heap h they want to move on (their visible heap or their hidden heap) and the number of objects i they wish to remove from that heap ($i \geq 1$). An action that removes i objects from the visible heap is invalid if $i > n_{\text{own}}$. Valid actions can be *successful* or *unsuccessful*: if players select the hidden heap and $i > n_{\text{opp}}$, the action is unsuccessful. If an action would result in state $(0, 0)$, the action is also unsuccessful. An unsuccessful action does not change the state of the board. If the action does not result in a winning or losing board, the opponent moves on the resulting board immediately afterwards according to their policy. If the opponent's move does not result in a winning or losing board either, the game continues with a new round of actions.
- Players receive a *reward* of r_u if their action is unsuccessful. Additionally, they receive a reward of r_w if their action yields a winning board, a reward of $-r_w$ if their opponent's action yields a winning board, and a reward of zero otherwise.
- Players receive an *observation* $(n_{\text{own}}, \text{succ}_{\text{own}}, \text{succ}_{\text{opp}})$ from each round – namely, the number of objects left in their visible heap, whether their own action succeeded, and whether their opponent's action succeeded. Players cannot observe their opponent's action itself.

PO-Nim is interesting for several reasons:

1) A player's strategy should depend on the strategy of the opponent: a good strategy must reason about the possible states of the game, but also about what the opponent believes the state of the game to be.
2) Removing objects from the hidden heap is useful to gather information about the state of the board, but it is also risky: we risk losing our turn if the action is unsuccessful, or giving away the number of objects left in our pile. A good strategy must trade off maximizing information gathering for one's own benefit and minimizing information gathering for the opponent.
3) If both policies are deterministic, the state transitions of the game and the observations of the game are deterministic. As a corollary of this fact, if the policy of the opponent is deterministic and known, we can infer the number of objects on the hidden heap and win the game.

In our work, we motivate and describe fully-nested interactive POMDPs and show how they can model PO-Nim

as a POMDP with an augmented state-space. Fully-nested I-POMDPs attempt to infer what policy their opponent is using to increase their chances of winning. We solve PO-Nim as a fully-nested I-POMDP by reducing the problem to a POMDP and solving the POMDP using SARSOP [4]. We show empirically that increasing the number of possible opposing policies does not render solving PO-Nim intractable.

II. Previous Work

Variants of nim have existed since ancient times. The first formal proof of a solution to nim for an arbitrary number of heaps and objects was published in 1901 by C. L. Bouton [1]. The optimal solution for a fully-observable game of nim involves making the move which results in a "nim-sum" of the board that is equal to zero. The nim-sum of the board is defined as the exclusive-or of the number of objects in each heap represented in binary. Because nim has been solved, we turn our attention to partially-observable nim, for which there is no clear strategy.

The concept of POMDPs that interact with other POMDPs has been the subject of active research in recent years. Problems in which a group of agents interact together in the face of uncertainty are common in fields like economics, robotics, or business. One popular framework for modelling these dynamics is the decentralized POMDP (Dec-POMDP). In Dec-POMDPs, agents seek to optimize a shared objective function with only a partial view of the environment [3]. Although very much related to our work, the context of Dec-POMDPs is strictly cooperative, and in PO-Nim, this is not the case.

Our approach builds on interactive POMDPs (I-POMDPs), which assume that the player is interacting with other I-POMDPs that may be seeking to optimize conflicting objectives [6]. I-POMDPs are solved recursively in levels, similarly to the logit level-k model in behavioral game theory. A level-0 (or "subintentional") agent is assumed to select actions without consideration for the opponent's belief or strategy. Commonly, the subintentional agent is assumed to select actions randomly. A level-1 I-POMDP is solved assuming the opponent is a subintentional agent. In finitely-nested I-POMDPs, a level-k I-POMDP is solved assuming the opponent is a level-(k-1) agent.

Finitely-nested I-POMDPs are not a good framework for PO-Nim due to the deterministic transition dynamics of the game. If the policy of the opponent is deterministic and known to be of level $k-1$, the opponent's moves on the hidden heap will be known and the hidden state of the board can be directly inferred, so the game can be solved trivially. It is also clear that the optimal strategy against a level-(k-1) opponent is not necessarily optimal for a level-(k-2) opponent (or any other lower-level opponent). Furthermore, it is not a good assumption that the level of the opponent will be known. A good model for solving PO-Nim should therefore incorporate some uncertainty over the level of the opponent, and have a mechanism for updating that belief based on observations about how they are playing.

III. Methods and Model

In this section, we introduce turn-based fully-nested I-POMDPs, explain how to reduce them to POMDPs, and show how to apply them to model PO-Nim. Just as in POMDPs, in fully-nested I-POMDPs, agents are uncertain about the current physical state of the environment. In the case of PO-Nim, agents are uncertain about how many objects remain in the hidden heap. In addition, in fully-nested I-POMDPs agents are uncertain about the level of their opponent. This is in contrast to traditional I-POMDPs, where agents assume the opponent plays at some fixed level. In fully-nested I-POMDPs, a level-k agent assumes that the opponent's level is one of 1 through $k-1$ but is uncertain about which. The agent will try to infer the level of the opponent based on some prior distribution over the possible levels of the opponent and their observations during the course of the game. Finally, agents are also uncertain about the belief of their opponent. Since the opponent is also assumed to be an I-POMDP, the belief of the opponent will be a belief over the current state of the game and the level and belief of *their* opponent.

A. Fully-nested I-POMDPs as POMDPs

The overall strategy we use for solving the fully-nested I-POMDP at a specific level is to first reduce the problem to a POMDP with an augmented state-space, and then apply one of several available offline POMDP solution algorithms to obtain a policy. The state s of the augmented POMDP for level k can be represented as (s_p, l, b), where s_p is the current physical state of the game, l is the level of the opponent ($l < k$) with associated policy π_l, and b is the opponent's current belief. The transition dynamics of the augmented POMDP are defined such that l does not change throughout the course of a game. The next augmented state depends on the current augmented state and the action a taken by the agent. Specifically, the next state will be a function of s_p, a, and $\pi_l(b)$, the action taken by the opponent. The new augmented state should update s_p but also b, which must be changed to reflect the opponent's updated belief after their observation. The reward and observation dynamics also depend on the joint action taken by both agents at a given augmented state.

In general, b is continuous. Since the state of the POMDP is augmented with b, it will also be continuous. Although POMDPs with continuous state-spaces are not intractable, they are generally harder to solve than their discrete counterparts. To get around this problem, we can describe the opponent's policy as a *finite-state controller*. A finite-state controller is an encoding of a policy as a finite-state machine where nodes are labeled with an action and edges are labeled with observations. An agent behaving according to finite-state controller executes the action at the current node and then transitions to a new node based on the observation they receive. This is similar to a tree representation of a conditional plan but it allows cycles, possibly avoiding an infinite number of nodes for an infinite-horizon plan. In a finite-state controller, nodes represent the possible beliefs that the agent will have about the state. The finite-state controller for PO-Nim against a random

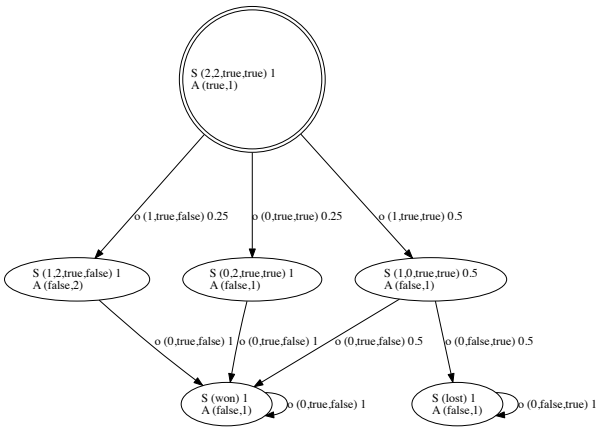

Fig. 1. Level-1 FSC for initial board $(2, 2)$ assuming the player goes first. Note that we do not need state augmentation to solve a level-1 I-POMDP since the opponent is known. Each node is labeled with the belief and the action that should be taken for that belief. Actions are labeled (h, i), where h = true indicates taking i objects from the visible pile and h = false from the opponent's pile. The belief is represented by the state of the board and the probability that the board is at that state. States are labeled $(n_{\text{own}}, n_{\text{opp}}, \text{succ}_{\text{own}}, \text{succ}_{\text{opp}})$ where succ_{own} indicates whether the player's last move succeeded and succ_{opp} whether the opponent's last move succeeded. Edges are labeled with the possible observations and their probabilities. Observations are represented as $(n_{\text{own}}, \text{succ}_{\text{own}}, \text{succ}_{\text{opp}})$. This policy has an expected reward of 0.5 against the level-0 subintentional opponent.

subintentional agent on a simple board with two objects in each heap is shown in Figure 1. Several POMDP solvers such as SARSOP can represent solutions to a POMDP as a finite-state controller. Since the opponent is itself a solution to a POMDP, its policy can be represented as a finite-state controller, and its current belief as one of a finite number of nodes in the controller. This allows us to solve the I-POMDP as a POMDP with a discrete state-space rather than a continuous one.

The size of the augmented state-space $\mathcal{S}$ of the POMDP grows as the sum of the number of nodes in each lower level's finite-state controller. Specificially, if $\mathcal{S}_p$ is the physical state space, then the size of the augmented state-space in the POMDP for a level-k fully-nested I-POMDP is

$$|\mathcal{S}| = |\mathcal{S}_p| \sum_{l=1}^{k-1} |\pi_l| \tag{1}$$

where $|\pi_l|$ is the number of nodes in the finite-state controller of a level-l solution. If the number of nodes in the controller is upper bounded by a constant, then the state space will only increase linearly with the level of the fully-nested I-POMDP. Frequently, however, the number of nodes in the controller will grow very quickly as the level of an I-POMDP is increased, even when there is no uncertainty over the level of the opponent. This has led to work on keeping the size of the controller bounded, such as bounded-policy iteration [6]. Incremental policy iteration can also help bound the size of a POMDP controller [2]. In the case of PO-Nim, we find that this is not necessary (see Section IV).

B. Turn-based I-POMDPs

In PO-Nim, the turn-based nature of the game necessitates an extension of our model. The transition dynamics of the game are dependent on the action of the agent but also on the action of the opposing agent *on the resulting board*. In other words, the transition between two augmented states includes an intermediate physical state which is the result of the first player's action and which the second player acts upon. Note that the optimal policy for the starting player may not be the same as the optimal policy for the player that goes second. For this reason, we solve policies at each level twice: once assuming the agent goes first, and again assuming the agent goes second. When the agent goes second, the transition dynamics of the game are modified such that the first "action" that the agent takes has no effect. This change means that the player that goes second will first observe the visible result of the opponent's first action before solving the rest of the game as if they go first. We summarize the turn-based transition dynamics for PO-Nim in Algorithm 1. The RESULT function represents the resulting physical state after taking the given action on the given physical state. In the case of PO-Nim, care must be taken to ensure that the physical state of the board is flipped for the opponent, and that the observations they receive are in line with their corresponding visible heap.

Algorithm 1 Deterministic transition function for a turn-based I-POMDP against an intentional opponent.

```
function TRANSITION(s, a)
    (s_p, l, b) ← s
    if go second and s_p is initial physical state then
        s''_p ← RESULT(s_p, π_l(b))
        b' ← b
    else
        s'_p ← RESULT(s_p, a)
        o ← OBSERVATION(s'_p)
        b' ← UPDATEBELIEF(b, o)
        s''_p ← RESULT(s'_p, π_l(b'))
    end if
    s' ← (s''_p, l, b')
    return s'
end function
```

The overall procedure for solving PO-Nim as a turn-based fully-nested I-POMDP is shown in Algorithm 2. We denote a level-k_1 agent as a level-k agent that goes first and a level-k_2 opponent as a level-k agent that goes second. We first solve level 1_1 against a random opponent that goes second and level 1_2 against a random opponent that goes first. We then solve level k_1 against a distribution over all levels l_2, and level k_2 against a distribution over all levels l_1, where $l < k$. For our purposes, we define the initial belief over the augmented states for each level as a uniform distribution over the augmented

Algorithm 2 Solving for the level-k fully-nested, turn-based I-POMDP

function SOLVE(k)
  $C_1 \leftarrow \{\pi_0^1\}$
  $C_2 \leftarrow \{\pi_0^2\}$
  for $l \leftarrow 1$ to k **do**
    $\pi_l^1 \leftarrow$ SOLVEAGAINST(C_2)
    $\pi_l^2 \leftarrow$ SOLVEAGAINST(C_1)
    $C_1 \leftarrow C_1 \cup \{\pi_l^1\}$
    $C_2 \leftarrow C_2 \cup \{\pi_l^2\}$
  end for
end function

states with $s_p = (N, N)$ and $l < k$, although this can easily be changed.

We use SARSOP to solve the POMDP at each level due to its ability to represent policies as finite-state controllers and good computational performance. Unlike QMDP [5], SARSOP does not struggle with the kind of information-gathering actions that are at the heart of a good PO-Nim strategy. Since we are using SARSOP, we must make one final adjustment to our model. In SARSOP, observation probabilities are conditioned on the action taken at a given timestep and the *next* state. However, in PO-Nim, whether or not an action succeeds depends on the current physical state as well. Therefore, we augment the state-space further with whether the previous action succeeded or not, as seen in Figure 1. After this change, the observation dynamics of the game are fully deterministic. Except for augmented states where the opponent is the level-0 subintentional model, the transition and reward dynamics are deterministic as well.

IV. RESULTS

To explore the computational feasibility of fully-nested I-POMDPs, we solve PO-Nim for various initial board configurations and levels. We will focus most of our discussion on an initial board with 3 objects in each heap ($N = 3$) because it is the smallest board where a good strategy is hard to reason about. Larger finite-state controllers for greater N are easy to encode computationally but difficult to illustrate neatly. We solve the I-POMDP for levels 1 through 15 as a POMDP with an augmented state space, as described in Section III. We use $r_{\mathrm{w}} = 10$ and $r_{\mathrm{u}} = -1$. The resulting finite-state controllers found by our method for levels 1_1, 1_2, 2_1, and 2_2 are included in the Appendix. A simulated game between levels 2_2 and 1_1 is shown in Figure 2. The corresponding transitions for this particular game are indicated in Figures 6 and 9 in the Appendix.

Figure 3 summarizes the expected reward of the fully-nested I-POMDP at different levels assuming all opponents at a lower level are equally likely. For low values of l, the expected reward grows as the level is increased because the probability of playing against the subintentional opponent is less likely; playing against a FSC, which is easier to beat, is more likely. For high values of l, the expected reward decreases as the level is increased because there is too much uncertainty about the level of the opponent and the resulting policy is no longer optimal for all lower-level opponents. The expected reward going down for high values of l does not mean that the quality of the policy necessarily decreases. Rather, it signifies that the prior distribution over the possible levels of the opponent is harder to beat. We also observed this trend for other values of N. At a given level, the expected reward for the player going first is generally lower than the expected reward for the player going second. This is expected because the nim-sum for two heaps and $N = 3$ is 0, so the first player would be at a disadvantage in the fully-observable version of the game.

```
        A B
Board: (3,3)
> A tries to take 1 from hidden heap
Board: (3,2)
> B observes (2,true,true)
> B tries to take 1 from hidden heap
Board: (2,2)
> A observes (2,true,true)
> A tries to take 3 from hidden heap
> Unsuccessful action!
Board: (2,2)
> B observes (2,true,false)
> B tries to take 2 from their heap
Board: (2,0)
> A observes (2,false,true)
> A tries to take 2 from their heap
> Unsuccessful action!
Board: (2,0)
> B observes (0,true,false)
> B tries to take 1 from hidden heap
Board: (1,0)
B wins! (-12.0, 10.0)
```

Fig. 2. Simulated game between solved finite-state controllers at levels 1_1 (player A) and 2_2 (player B). 2_2 wins with a reward of 10, and 1_2 loses with a reward of -12.

In simulations between controllers, we observe that in some cases the solved policy incurs a penalty of r_u in situations where this is guaranteed to happen in order to obtain information about the level of the opponent and have a chance at beating it. If we set to $r_u = 0$, solved controllers begin the game by removing a single object from their opponent's heap, after which action (false, N) is guaranteed to be unsuccessful and effectively becomes a "pass" action at no cost. This results in opponents repeatedly passing back and forth until one of them concludes (correctly or otherwise) the level of their opponent. To win, a level k opponent passes one more time than the number of times that the opponent at level $k-1$ passes. Since opponents at lower levels cannot plan against a higher-level opponent, they incorrectly conclude the level of the opponent and make a move, after which the higher-level opponent can win the game. This results in level-k models beating all lower levels except the subintentional model deterministically. By introducing a penalty $r_u < 0$ for unsuccessful moves, passing back and forth becomes suboptimal.

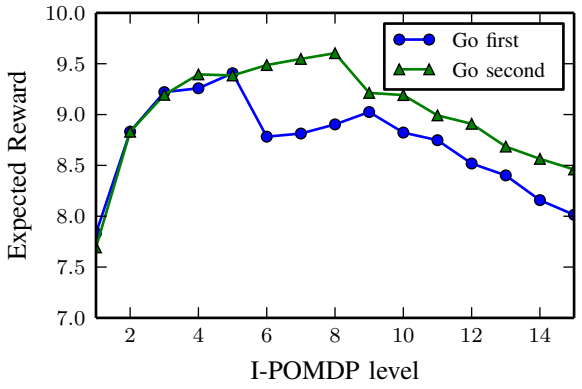

Fig. 3. Expected reward of I-POMDP solution in a PO-Nim game with $N = 3$, $r_w = 10$, and $r_u = -1$. Assumes all opponents at a lower level are equally likely.

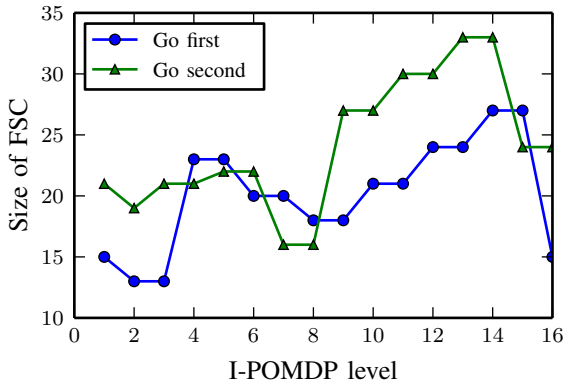

Fig. 5. Number of nodes in the finite-state controller of the I-POMDP in the solution produced by SARSOP for $N = 3$.

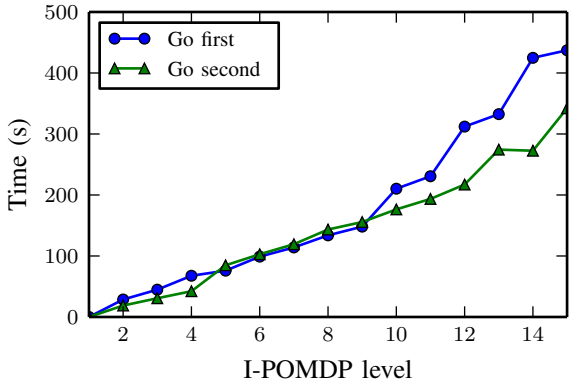

Fig. 4. Time taken to obtain the finite-state controller for each level using SARSOP given the finite-state controllers for all lower levels have already been compute ($N = 3$). Includes initialization time.

We found that solving PO-Nim did not become intractable as the level of the fully-nested I-POMDP was increased. As shown in Figure 4, the time taken by SARSOP to increment the level of the I-POMDP by solving an additional augmented POMDP varies roughly linearly with the level. This is likely due to the number of nodes in the finite-state controller of the solution not growing very quickly as the level is increased. We illustrate the size of the finite-state controller of the solved I-POMDP in Figure 5. The number of nodes in the finite-state controller sometimes decreases, keeping its size from growing out of control. This is likely a property of the deterministic dynamics of PO-Nim. SARSOP solved the POMDPs at each level exactly (with policies having a utility within 10^{-7} of the optimal value), also likely due to the simple nature of PO-Nim. The final number of alpha vectors correlated with the size of the controllers and ranged from 12 for level 2_1 to 61 for level 13_2.

V. Conclusion

We presented a framework for solving turn-based, partially-observable games as fully-nested interactive POMDPs. Fully-nested I-POMDPs do not require assuming that the opponent plays at a fixed level. Instead, they allow us to solve for an optimal policy against an arbitrary prior distribution over the possible levels of the opponent. We motivated fully-nested I-POMDPs by introducing PO-Nim, an example of a game where it is not reasonable to assume the level of the opponent a-priori. We showed how to reduce PO-Nim to a POMDP with an augmented state space, and explained how to tackle the turn-based nature of the game. We showed experimentally that it is possible to solve a level-k fully-nested PO-Nim I-POMDP using SARSOP for relatively large values of k.

Extensions to this work could formally explore which types of games yield small finite-state controllers without having to resort to more sophisticated solver techniques like interactive or bounded policy iteration. On the applied side, more complex turn-based games with partial observability such as Stratego or Battleship offer similar dynamics to PO-Nim with a much larger physical state-space. It is likely that other techniques would be required to solve either game as a fully-nested I-POMDP.

In solving a level-k I-POMDP, we defined the initial belief over the level of the opponent as a uniform distribution over all lower-level opponents. It would be interesting to experimentally obtain a good prior over the possible levels of a human opponent. This could be done by recording a large number of human PO-Nim games and leveraging our solved level-k policies to fit the data to a distribution over k by using Bayesian parameter learning.

A third extension to our work would involve formalizing I-POMDPs with stochastic policies. Playing against the subintentional level-0 is difficult because state transitions are not determinsic. If we allow for probabilistic actions at each belief for intentional levels, we may be able to develop policies that higher-level opponents find harder to beat.

Acknowledgment

The author would like to thank Ekhlas Sonu and Zachary Sunberg for their helpful ideas and suggestions in the early stages of this project.

REFERENCES

[1] C. L. Bouton. *Nim, a game with a complete mathematical theory.* The Annals of Mathematics 3.1/4 (1901): 35-39.

[2] M. Grzes and P. Poupart. *Incremental policy iteration with guaranteed escape from local optima in pomdp planning.* Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[3] M. Kochenderfer. *Decision Making Under Uncertainty.* The MIT Press (2015), pp 159–187.

[4] H. Kurniawati, D. Hsu, and W. S. Lee. *SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces.* Robotics: Science and Systems, Vol. 2008. 2008.

[5] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. *Learning policies for partially observable environments: Scaling up.* Machine Learning Proceedings 1995: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9-12, 1995, p 362. Morgan Kaufmann.

[6] E. Sonu and P. Doshi, *Scalable solutions of interactive POMDPs using generalized and bounded policy iteration*. Autonomous Agents and Multi-Agent Systems, May 2015, Volume 29, Issue 3, pp 455-494.

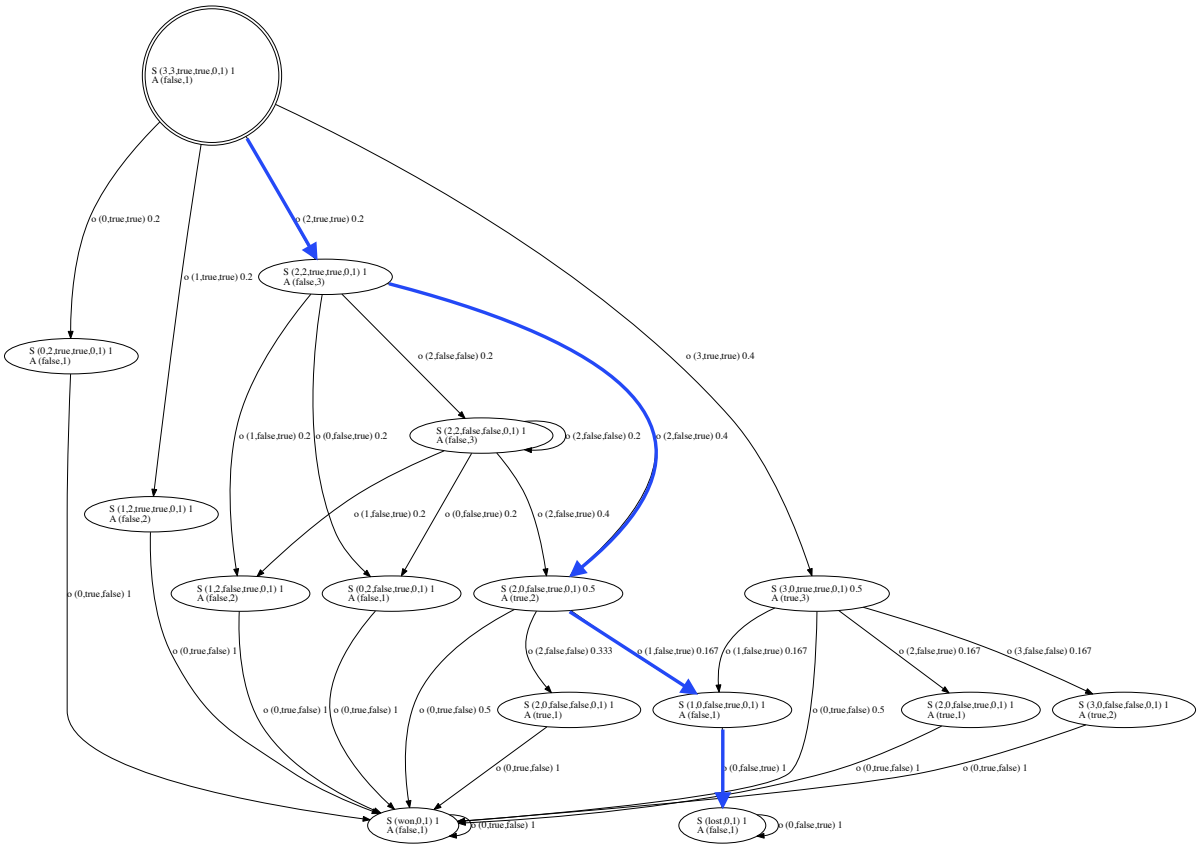

Fig. 6. Finite-state controller for level 1_1 on a $N = 3$ board. At level 1_1, the opponent is assumed to be 0_2. Transitions are highlighted for a losing game against the FSC for level 2_2, shown in Figure 2. States are labeled $(n_{\text{own}}, n_{\text{opp}}, \text{succ}_{\text{own}}, \text{succ}_{\text{opp}}, l, b)$ where succ_{own} indicates whether the player's last move succeeded, succ_{opp} whether the opponent's last move succeeded, l is the level of the opponent, and b is the opponent's belief as a discrete index into their finite-state controller. Actions and observations are labeled as in Figure 1.

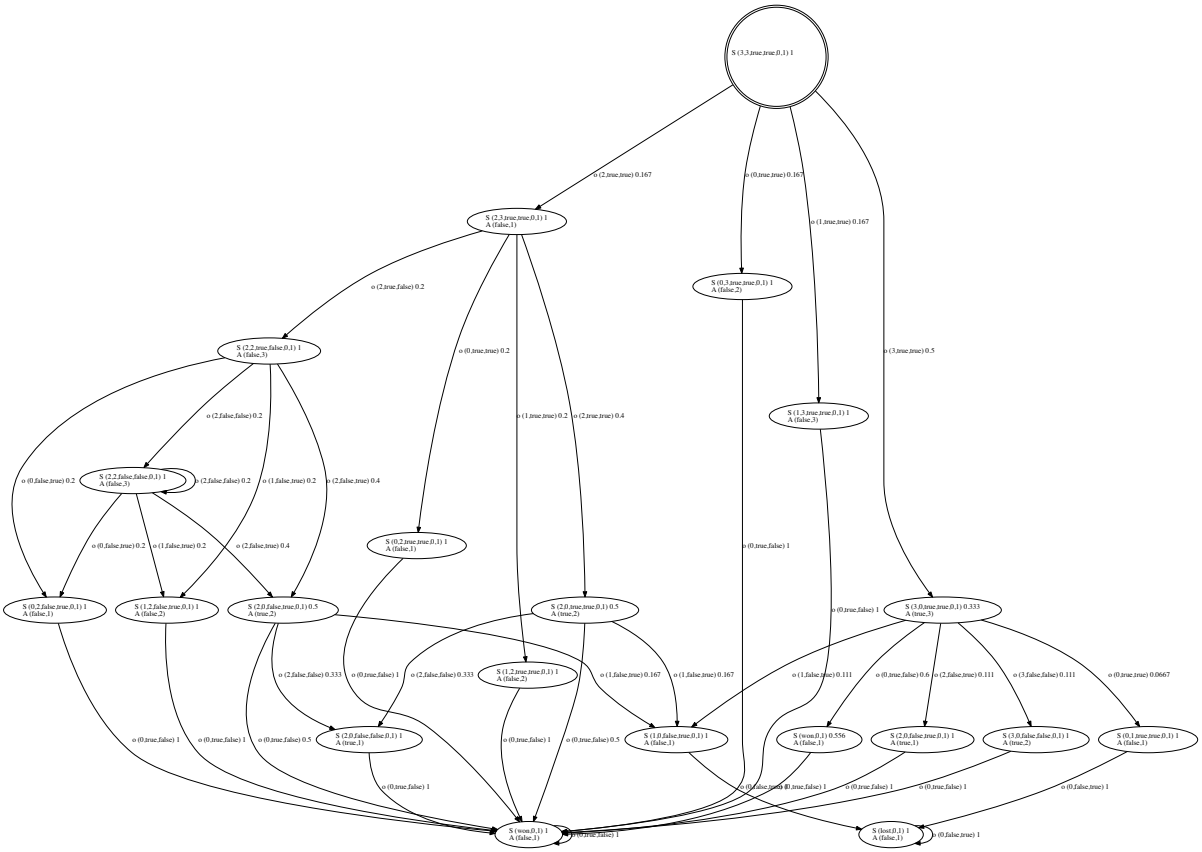

Fig. 7. Finite-state controller for level 1_2 on a $N = 3$ board. The opponent is 0_1. The initial state is a dummy state: we cannot act until 0_1 has moved, at which point we receive the first observation. Beliefs, actions, and observations are labeled as in Figure 6.

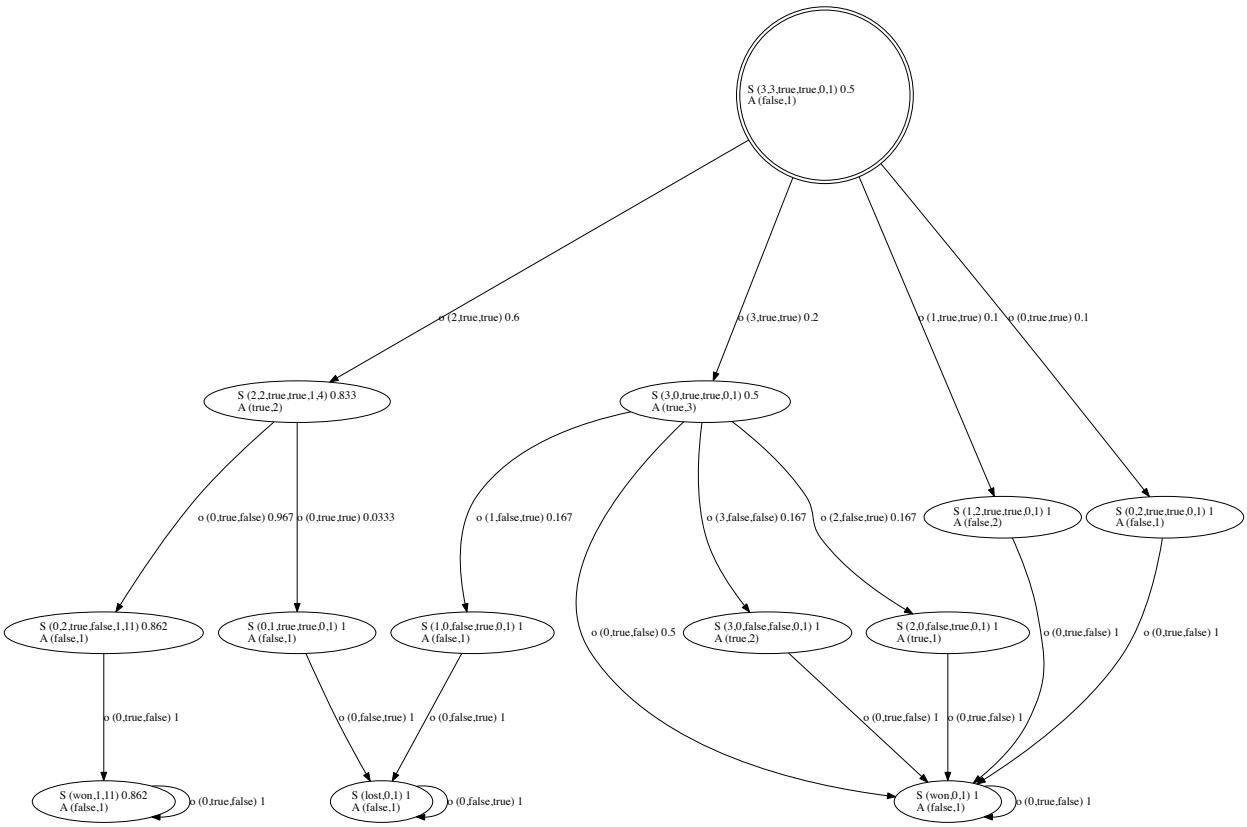

Fig. 8. Finite-state controller for level 2_1 on a $N = 3$ board. The opponent is either 0_2 or 1_2 with equal probability. Beliefs, actions, and observations are labeled as in Figure 6.

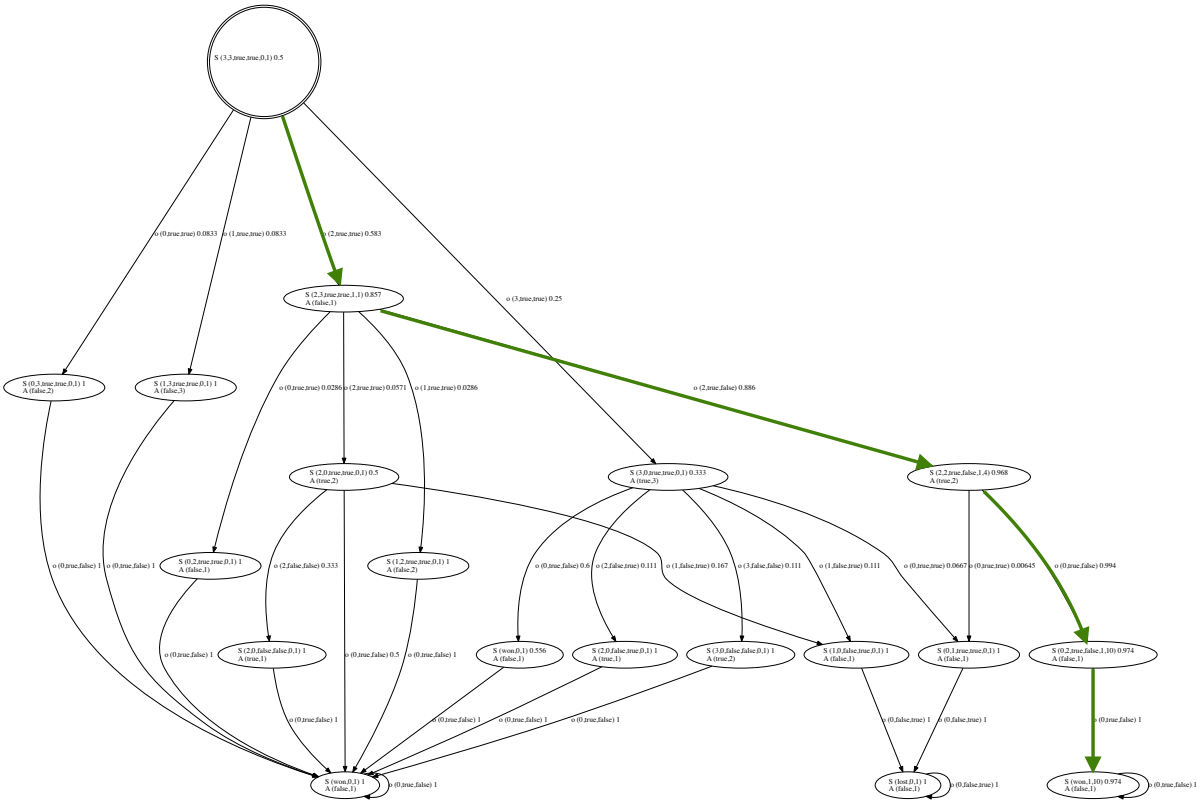

Fig. 9. Finite-state controller for level 2_2 on a $N = 3$ board. The opponent is either 0_1 or 1_1 with equal probability. The initial state is a dummy state: we cannot act until the opponent has moved, at which point we receive the first observation. Transitions are highlighted for a winning game against the FSC for level 1_1, shown in Figure 2. Beliefs, actions, and observations are labeled as in Figure 6.