

# Stochastic Games for Interactive Manipulation Domains

Karan Muvvala<sup>\*1</sup>, Andrew M. Wells<sup>\*2</sup>, Morteza Lahijanian<sup>1</sup>, Lydia E. Kavraki<sup>2</sup>, and Moshe Y. Vardi<sup>2</sup>

**Abstract**—As robots become more prevalent, the complexity of robot-robot, robot-human, and robot-environment interactions increases. In these interactions, a robot needs to consider not only the effects of its own actions, but also the effects of other agents’ actions and the possible interactions between agents. Previous works have considered reactive synthesis, where the human/environment is modeled as a deterministic, adversarial agent; as well as probabilistic synthesis, where the human/environment is modeled via a Markov chain. While they provide strong theoretical frameworks, there are still many aspects of human-robot interaction that cannot be fully expressed and many assumptions that must be made in each model. In this work, we propose *stochastic games* as a general model for human-robot interaction, which subsumes the expressivity of all previous representations. In addition, it allows us to make fewer modeling assumptions and leads to more natural and powerful models of interaction. We introduce the semantics of this abstraction and show how existing tools can be utilized to synthesize strategies to achieve complex tasks with guarantees. Further, we discuss the current computational limitations and improve the scalability by two orders of magnitude by a new way of constructing models for PRISM-games.

## I. INTRODUCTION

Traditionally, robots have accomplished complex tasks through *planning* i.e., computing a “path” from the initial state to a goal state. As robots become more prevalent, the complexity of robot-robot, robot-human, or robot-environment interactions increases. In these interactions, a robot needs to consider not only the effects of its own actions, but also the effects of other agents’ actions as well as the possible interactions between agents. These complexities mean *planning* is often insufficient. Instead the robots should compute a *strategy*, anticipating the possible effects of each agent’s actions and reasoning in advance how it should respond to different contingencies.

In order to ensure safety and general correctness, synthesis, either reactive or probabilistic, has emerged as a promising approach to creating correct-by-construction robot strategies [1]–[3]. In reactive synthesis, the worst-case behavior of the human/environment is considered. This is overly conservative and lacks the power to describe many scenarios where one possibility is known to be more likely than another. Essentially, by assuming the human/environment is *adversarial*, the

This work was supported in part by NASA 80NSSC21K1031, NASA 80NSSC17K0162, NSF 1830549, and NSF RI 2008720. Authors would also like to thank the authors of PRISM, especially Dr. Dave Parker for their excellent tool and for their help in modifying it to import stochastic games.

<sup>\*</sup>Equal contribution

<sup>1</sup>Aerospace Eng. Sciences Dept. at the University of Colorado Boulder. [firstname.lastname@colorado.edu](mailto:firstname.lastname@colorado.edu)

<sup>2</sup>Computer Science Dept. at Rice University. Andrew Wells was a student at Rice University at the time this work was conducted. [andrewmw94@gmail.com](mailto:andrewmw94@gmail.com), [{kavraki,vardi}@cs.rice.edu](mailto:{kavraki,vardi}@cs.rice.edu)

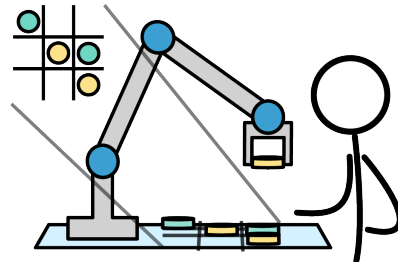


Fig. 1: Tic-tac-toe game between a robot and a human. The robot is unaware of the level of expertise of the human and suffers from the “trembling hand” problem. In this case, the robot needs to reason about the probabilities of reaching a given state as well as the strategic responses of both agents from that state.

robot becomes overly pessimistic, resulting in conservative strategies that are “unfriendly” and “competetive” with the human. It also drastically limits the scenarios, for which task-completion guarantees can be provided. This is the reason previous methods place unwieldy limitations on human [3], e.g., human takes at most a fixed number of actions.

Probabilistic synthesis has been proposed as an alternative to reactive synthesis to address this issue. Those methods view the human/environment as a *probabilistic* agent [4], [5]. This can describe many types of human behaviors, not just adversarial behavior; however, it is unrealistic as it assumes that the human behavior is Markovian. Generally, humans have their own objectives and take actions accordingly. Hence, they should be treated as strategic agents. For instance, in a tic-tac-toe game, as depicted in Fig. 1, both human and robot aim to win. However, it is not clear if the human is a novice player (makes imperfect moves), a master player (makes perfect moves), or somewhere in between. Hence, both probabilistic and strategic aspects are present, which a purely probabilistic model cannot capture [5], [6].

To address these limitations, we present robot strategy synthesis using stochastic games. Intuitively, stochastic games can be considered as a generalization of Markov Decision Processes (MDPs), where instead of one agent making decisions, multiple agents make decisions. Stochastic games subsume the expressive power of reactive and probabilistic synthesis; giving us the most general model for robot strategy synthesis. For instance, they allow modeling of the scenario in Fig. 1, even for a robot with imperfect actuation, e.g., may accidentally drop a piece in an unintended location due to a bad grasp. The key benefit of stochastic games compared to prior works is not that the models are more accurate (though this can be the case), but rather modeling human-robot

manipulation as a stochastic game makes fewer assumptions and is thus more robust. Because of their expressive power, however, stochastic games bring new challenges in terms of scalability. This paper only begins to address these challenges.

In this work, we bridge the gap between robotic manipulation domain and the expressive power of stochastic games. We mainly focus on the abstraction construction of the continuous manipulation domain in the presence of a human and robot action uncertainty as a discrete two-player stochastic game. We present conditions and semantics under which this abstraction can be viewed as a turn-based game, improving computation tractability. Further, we show that the strong assumption that the human takes a pre-defined number of actions (as in [7]) can be relaxed in our abstraction. We also provide an implementation that enables scalability by bypassing the built-in model construction of stochastic games in the existing tool, namely PRISM-games [8]. Finally, we illustrate the power of our approach on several case studies and show scalability in a set of benchmarks.

The contributions of this work are fourfold: (i) we formalize how to model the human-robot manipulation domain as turn-based, two-player stochastic game and use existing tools to synthesize optimal strategies for the robot; (ii) we relax the assumptions on human interventions while still treating the human as a strategic agent; (iii) we improve the scalability of the existing tool and provide an open-source tool for efficient synthesis for robotic manipulation scenarios (available on Github [9]); (iv) we illustrate the efficacy of our proposed approach on several case studies and benchmarks.

**Related Work.** *Synthesis* is the problem of automatically generating a correct-by-construction plan or strategy from a high-level description (specification) of a task. The specifications are usually expressed in Linear Temporal Logic (LTL) [10], and for robotic systems, LTL interpreted over finite traces (LTL<sub>f</sub>) [11], [12] is popular due to its ability to describe tasks that need to be completed in finite time. When an agent interacts with the world, we are interested in synthesizing a strategy that *reacts* to the environment. Reactive synthesis has been examined as a stand-alone problem as well as in robotics [1], [2]. Most works on reactive synthesis for robotics focus on mobile robots [1], [13]–[15], which has a relatively simple state space compared to manipulation. Reactive synthesis has also been examined for manipulation [7], [16], [17] domains. We build on these later works in this paper.

Probabilistic synthesis has been examined for general domains [18]–[22], including robotic manipulation [5]. In the context of learning, stochastic games with unknown transitions have been studied for abstracted robotic systems [23]. In synthesis, we assume the transitions between states are known a priori. Existing works on stochastic synthesis for manipulation domain use MDPs, which only allow one strategic agent. Thus, they assume the human behaves in a mechanical fashion and synthesize an optimal robot policy. Using stochastic games allows us to reason about a strategic human agent. We focus on modeling human-robot manipulation scenarios with stochastic games where tasks are defined using formal language, which has not been studied.

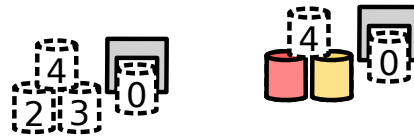


Fig. 2: Manipulation domain: (left) the locations of interest, where the Else location ( $L_1$ ) contains all objects not otherwise shown. (right) Initial state with red and yellow blocks at  $L_2$  and  $L_3$  and the blue block at  $L_1$ .

## II. PROBLEM SETUP

In this work, we focus on a robotic manipulator with “trembling hands” operating in the presence of a human. Given a high-level task for the robot and knowledge on the behaviors of general humans, our aim is to synthesize a strategy for the robot to maximize the probability of completing the task.

### A. Probabilistic Abstraction of Manipulation Domain

We model the manipulation domain as an MDP by abstracting configuration space  $C = C_r \times C_o$ , where  $C_r$  and  $C_o$  are the robot and movable objects configuration spaces, respectively. Intuitively, a state of this MDP captures relevant features of  $C$ . That is, the state is a tuple of objects and their locations. Further, using Planning Domain Definition Language (PDDL) [24], we ground and define robot actions along with preconditions and effects of these actions from every state [17]. Since our actions have stochastic outcomes, we define a probability distribution associated with the effects of robot actions. Formally,

**Definition 1** (Probabilistic Manipulation Domain Abstraction). *A probabilistic manipulation domain is an MDP tuple  $\mathcal{M} = (S, A, P, s_0, AP, L)$  where,*

- $S$  is a finite set of states,
- $s_0 \in S$  is the initial state,
- $A$  is a finite set of robot actions,
- $P : S \times A \times S \rightarrow [0, 1]$  is the probability distribution over the effects of the robot’s action  $a \in A$  and  $\sum_{s' \in S} P(s, a, s') = 1$  for all state-action pairs,
- $AP$  is the set of task-related propositions that can either be true or false, and
- $L : S \rightarrow 2^{AP}$  is the labeling function that maps each state to a set of  $AP$  that are true in  $s \in S$ .

An execution of MDP  $\mathcal{M}$  is a path  $\omega = \omega_0 \xrightarrow{a_0} \omega_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} \omega_n$ , where  $\omega_i \in S$ ,  $a_i \in A$ ,  $\omega_0 = s_0$ , and  $P(\omega_i, a_i, \omega_{i+1}) > 0$  for all  $0 \leq i \leq n - 1$ . The set of finite paths is denoted by  $S^*$ . The *observation trace* of  $\omega$  is  $\rho = L(\omega_0) \dots L(\omega_n)$  the sequence of sets of atomic propositions observed along the way. We define the task of the robot according to these observations, below. A *robot strategy*  $\pi : S^* \rightarrow A$  is a function that chooses an  $a \in A$  for the robot given the path  $\omega \in S^*$  executed so far.

**Example 1** (MDP). *Consider the continuous manipulation domain in Fig. 2. The corresponding MDP is depicted in Fig. 3. The robot is tasked with building an arch with blue block (not shown) on top. The initial state is defined as*

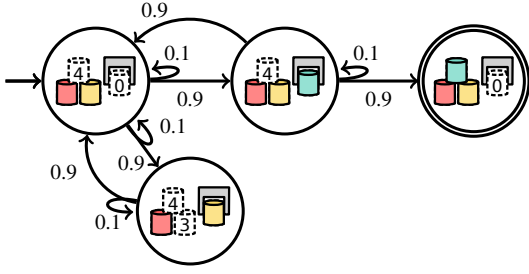


Fig. 3: Example abstraction of manipulation domain from Fig. 2 with stochasticity for robot actions.

$s_0 := \{O_{02}, O_{13}, O_{21}\}$  where  $O_{ij}$  corresponds to object  $i$  placed at location  $j$ . Here  $O_0, O_1, O_2$  are the red, yellow, and blue blocks, respectively. From the initial state, under the robot-grasp blue block action, there is a 10% chance of failure and a 90% chance of success. The alternate action is to grasp the yellow block and finally place blue on top.

### B. Manipulation Tasks as LTL<sub>f</sub> formulas

As robotic tasks must be accomplished in finite time, Linear Temporal Logic over finite traces (LTL<sub>f</sub>) [11] is an appropriate choice for the specification language. That is because LTL<sub>f</sub> is very expressive (same syntax as LTL) but its interpretations are over finite behaviors.

**Definition 2** (LTL<sub>f</sub> Syntax). *Given a set of atomic propositions AP, an LTL<sub>f</sub> formula is defined recursively as*

$$\phi := \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid \phi U \phi$$

where  $p \in AP$  is an atomic proposition,  $\top$  (“true”),  $\neg$  (“negation”) and  $\wedge$  (“conjunction”) are the Boolean operators, and  $X$  (“next”) and  $U$  (“until”) are the temporal operators.

The common temporal operators “eventually” ( $F$ ) and “globally” ( $G$ ) are defined as:  $F\phi = \top U \phi$  and  $G\phi = \neg F \neg\phi$ .

The semantics of an LTL<sub>f</sub> formula are defined over finite traces in  $(2^{AP})^*$  [25]. We say a path  $\omega \in S^*$  accomplishes  $\phi$ , denoted by  $\omega \models \phi$ , if its observation trace satisfies  $\phi$ .

**Example 2** (Example LTL<sub>f</sub> specification). *The LTL<sub>f</sub> formula for constructing an arch from Fig. 2 with any two blocks as support and blue block on top can be written as,*

$$\phi_{\text{arch}} = F (p_{\text{block, support}_1} \wedge p_{\text{block, support}_2} \wedge p_{\text{blue, top}}) \wedge G (\neg(p_{\text{block, support}_1} \wedge p_{\text{block, support}_2}) \rightarrow \neg p_{\text{blue, top}})$$

where  $\text{support}_i \in \{L_2, L_3\} \forall i \in \{1, 2\}$  and  $\text{top} := L_4$ .

### C. Problem Statement

In this work, we are interested in synthesizing strategies for the robot operating in the presence of human. In our setting, human behavior can be abstracted as human moving objects and the actions can be formalized as object(s) moving from one location to another. We aim to develop a general framework; hence, we do not assume knowledge about the particular human, with whom the robot is interacting.

Further, we assume that humans are strategic agents who choose actions according to some objective, which is latent

to the robot. Also, we assume general knowledge on the likelihood of the human moving a specific block to some location. Such general likelihoods can be inferred from past experiences (data) on various humans. Our goal is to synthesize a strategy for the robot to maximize the likelihood of achieving its task.

**Problem 1.** *Given a robotic manipulator with its MDP abstraction and LTL<sub>f</sub> task formula  $\phi$  in the presence of a human with a latent objective and general likelihood of (taking) actions,*

- 1) *Abstraction: generate a finite abstraction of the interaction between the robot and human through object manipulation such that it captures the strategic and stochastic aspects of both agents,*
- 2) *Synthesis: synthesize a strategy for the robot that maximizes its probability of accomplishing  $\phi$ .*

There are several challenges in Problem 1. We want to model a strategic human with imperfect decision making capabilities. Also, we want to allow the robot to be a strategic agent that could fail sometimes in executing its action. Hence, the focus of our approach is the construction of an abstraction that captures all the necessary aspects of the problem for best decision making. Additionally, note that both the manipulation domain and reactive synthesis are notorious for their *state-explosion* problem [5], [17]. Problem 1 combines the two; hence, computational tractability is an aspect that we want to ensure in our approach.

## III. STOCHASTIC GAME ABSTRACTION

In this section, we discuss how strategic and stochastic elements of the human and robot are combined to form a two-player stochastic game. Specifically, we deal with a fully observable two-player game. Naturally, this game is concurrent in the continuous domain, but concurrent games are known to suffer from computational tractability [26]. Our goal is to define semantics that allow a turn-based modeling for the purpose of strategy synthesis such that the execution of the strategy is seemingly concurrent at the runtime. We first formally define a two-player, turn-based stochastic game (simply, *stochastic game*) [27], and then show our abstraction to this game.

**Definition 3** (Stochastic Game). *A stochastic game is a tuple  $G = (S, s_0, A_s, A_e, T, C, AP, L)$ , where  $S, s_0, AP$  and  $L$  are as in Def. 1, and*

- $A_s$  and  $A_e$  are the finite set of robot & human actions,
- $T : S \times (A_s \cup A_e) \times S \rightarrow [0, 1]$  is the probabilistic transition relation, and
- $C : S \mapsto \{s, e\}$  designates which player controls the choice of action at each state.

Here, players  $s$  (system) and  $e$  (environment) are the robot and human, respectively. An execution of the game  $G$  is a sequence of visited states as players take turns in making moves. The choice of action for Player  $i \in \{s, e\}$  is determined by the strategy  $\pi_i : S^* \rightarrow A_i$  that picks actions according to the execution of the game so far.



For the strategic players, we follow the models of prior work [3]. The robot player’s actions follow a standard pick-place domain, which typically can be encoded in PDDL as described above. The human player has the same abilities but is assumed to move relatively quickly compared to the robot. Additionally, unlike previous approaches, we do allow the human to hold onto an object. Thus, we model the robot’s gripper and the human gripper and make a fairness assumption that the human will eventually return the object.

**Game States.** As in [3], we model the continuous world by grouping locations into “regions of interest”. These include a “end-effector” region representing the robot’s gripper and an “Else” region representing all locations not particularly specified. To allow the robot to react at any point, the model should be constructed such that every valid arrangement of objects in the real world has game states for both human and robot turns. These states are equivalent to the robot MDP states in Def. 1.

**Game Actions.** In prior works [7], [17], [28], the human is typically assumed to move faster than the robot, leading to multiple human moves per robot move. We follow this assumption and examine several models of turn allocation. We have several modeling choices and present results for all of them. The set of robot actions  $A_s = A$  is the same set of actions in Def. 1. The human actions  $A_e$  are every possible move of the objects to the locations of interest, “Else”, and human’s gripper. Then, the transition relation  $T(s, a, s') = P(s, a, s')$  if  $a \in A_s$ ; otherwise, it is obtained from the likelihood of the human actions as discussed below.

In reactive synthesis [7], a limit  $k$  is placed on the total number of human interventions to ensure the specification is realizable. This limit is unintuitive and somewhat unrealistic. For example, suppose the model assumes the human intervenes at most 30 times ( $k = 30$ ). Then, during execution, once the robot observes the 30th action, it will act as though the human will no longer interfere. Unfortunately, unless there is some external reason for this limit, the robot should arguably assume the human is *more* likely to interfere because it has observed this happen 30 times already.

We generalize this as a *ratio* of human and robot actions. For example, we could allow 1 human action for every 2 robot actions (denoted by 2 : 1). We implement this using counters that reset every time the game changes control from one player to another. So a player cannot “skip” turns now in order to take more consecutive turns later. While this could be encoded as a two-player game, the ability to express the stochasticity in robot’s success rate (of executing actions) or/and the human’s tendency to intervene at particular locations can not modeled using a purely game theoretic approach.

Our other model uses a probability of handing control from one player to another. This implies a probabilistic limit but avoids setting a hard limit on action for either player. This is achieved by including an action in  $A_e$  that evolves to a state after which the human does not intervene. This is a natural weakening of the hard limit on human intervention. Note that this cannot be modeled using MDP where the effects of human actions are inherently random in nature. Thus,

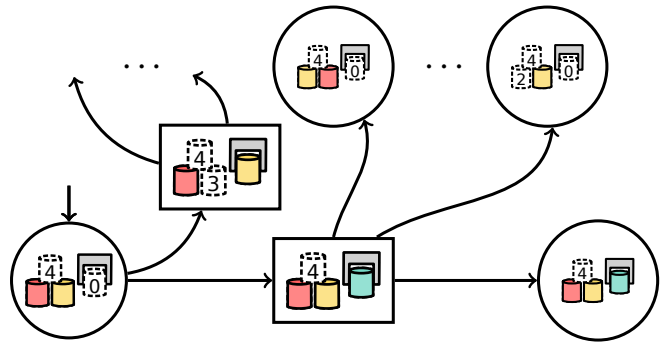


Fig. 4: Stochastic game variant of MDP in Fig. 3. The circle and rectangle states belong to the robot and human player. For this e.g. we allow human to move objects from the robot’s gripper. The top row shows multiple human movements, while the state on the right corresponds to no human intervention.

stochastic games allows us to reason over strategic players while also considering stochasticity in their execution.

In addition to assigning control of game states to each player, we need some way to turn the continuous, real-time interaction into a turn-based game. We do this as in [7], by assuming certain robot actions are “atomic” while giving the human actions priority over all non-atomic actions. Here, as in previous papers, the atomic robot actions are grasp and place (not including the transit or transfer preceding the opening / closing of the gripper). Once we have chosen a way to model actions, and under our assumptions about discrete states and atomic action executions, we can create a turn-based stochastic game.

**Example 3.** A partial two-player stochastic game for our manipulation domain is shown in Fig. 4. The actions taken in circular states are controlled by the system and those taken in rectangular states are controlled by the environment. The same action could stochastically lead to different possible states. For example, the robot’s action from initial state is to grasp a block from the initial state and stochastically determine whether to grasp the yellow or the blue block.

**Remark 1.** Winning the game translates to finishing task  $\phi$ , and winning strategies are strategies that guarantee task completion. Termination of the game is typically defined as reaching an accepting or violating finite prefix of a trace. This could be insufficient in certain cases, e.g., a robot asked to tidy a room will stop once the room is cleaned even if the human is approaching some object to displace it. Our game modeling allows the human and robot to “negotiate” termination so that the robot only considers the task complete when the human agrees.

#### IV. STRATEGY SYNTHESIS

For a given  $LTL_f$  specification, synthesis reduces to solving a stochastic game for a reachability objective, i.e., reach a target state [8]. That game is the composition of  $G$  with the automaton that is constructed from  $\phi$  [11]. Solving stochastic games with reachability objectives lies in the complexity class  $NP \cap coNP$  [29]. PRISM-games makes use of the model

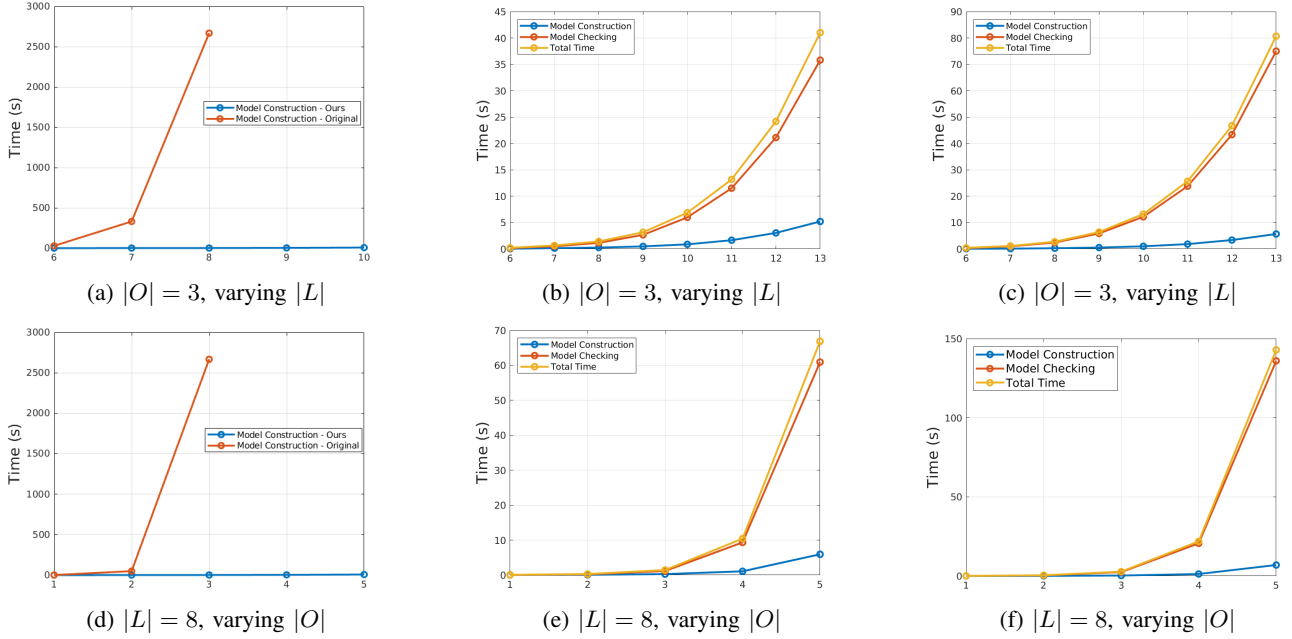


Fig. 5: Benchmark results for different scenarios using our approach. (a) and (d) illustrate the model construction time using the original PRISM-games and our implementation for the probabilistic human termination scenario. (b) and (e) illustrate computation times for the 1:1 action ratio scenario, and (c) and (f) correspond to the probabilistic human termination scenario.

checking algorithms described in [8], that relies on Value Iteration to compute the values for all states of the game [30]. The algorithm can be decoupled into two stages.

*Precomputation Stage:* During this stage, we identify states of the game for which the probability of satisfaction is 0 or 1, and the maximal end components of the game. Informally, an end component is a set of states for which, there exists a robot strategy such that it is possible to remain forever in that set once entered. Efficiency and accuracy can be improved by using this step. Next, numerical computation is performed on the remaining states in the game.

*Numerical computation stage:* The probability of reaching a target state is 1 if the state belongs to the target end component, else we iteratively update state values until we reach a fixed point. At every iteration, we perform  $\max_a(\sum_{s'} T(s, a, s') \cdot p(s'))$  if  $s$  belongs to robot player else we perform  $\min_a(\sum_{s'} T(s, a, s') \cdot p(s'))$ . Here  $T$  is from Def. 3, and  $s, a, s'$  are the current state, action, and successor state, respectively.  $p(s')$  denotes the value associated with the successor state in the previous iteration. While PRISM-games is a mature toolbox, the implementation for solving stochastic games is less mature than tooling for MDPs, and we found the bottleneck to be model construction rather than Value Iteration. Below, we discuss how we mitigate this bottleneck.

## V. IMPLEMENTATION AND RESULTS

Here, we present benchmarks based on experiments from [5]. We run our tests using PRISM-games [8] and discuss our modifications to remove a performance bottleneck when importing models. All the experiments are run on an Intel i5 -13th Gen 3.5 GHz CPU with 32 GB RAM. The tool is

available on GitHub [9]. The results are shown in Fig. 5.

**Scalability.** We test a simple pick-and-place manipulation domain, varying the number of objects and locations. Three locations are reserved for the robot and human gripper, and the terminal location for each player. Only one object can be manipulated by the robot and human, while multiple objects can be placed at other locations. The task is to place objects in their desired locations.

PRISM’s default configuration reads in modeling files written in the PRISM modeling language. As PRISM is (primarily) a symbolic engine, a structured, hierarchical model is preferred as it exploits regularity in the abstraction. Our model is naturally flat, and hence PRISM modeling language suffers from scalability. Therefore, we implement functionality to import the stochastic games models through the direct specification of their transition matrix, state, label, and player vectors.

We use a Python script to automate the construction of the model files for direct specification of transition matrix and state vectors outside PRISM and then import them in PRISM-games. We benchmark this method of model construction as shown in Fig. 5a and Fig. 5d. We see that while the PRISM’s original implementation (in red) fails to scale beyond 3 objects and 8 locations, our approach (in blue) not only scales beyond this bottleneck but is also 2 orders of magnitude faster.

We also present benchmarks based on the modeling choices described in Sec. III. Fig. 5b and Fig. 5e correspond to model construction and synthesis time for 1:1 scenario where we allow one human action for every robot action. In this scenario, the human could potentially undo every action the robot does and hence the robot cannot guarantee

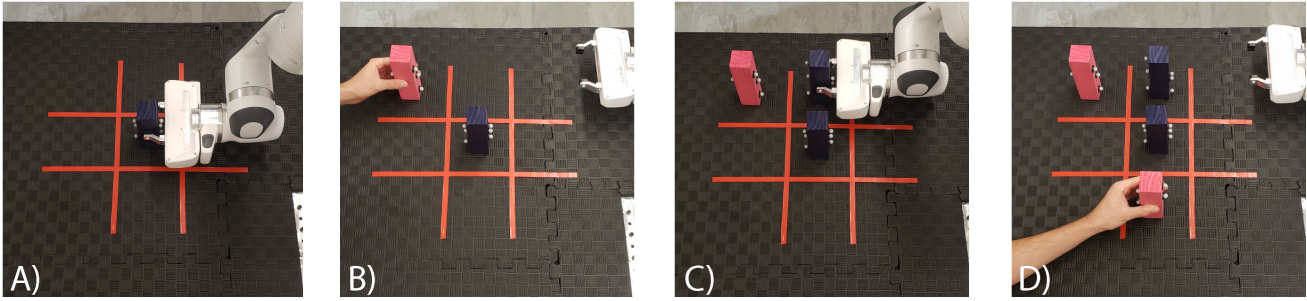


Fig. 6: The game begins with (A) and (B). In state (C) the robot chooses a move that maximizes the probability of human failure under the “trembling hand” model. In (D) the human will likely place the object in the bottom center, but has two open neighbor cells. Under the robot strategy, the human will have three more chances to fail (video: <https://youtu.be/UUBW7QEw6Ng>).

TABLE I: Abstraction & Synthesis comp. times for 3 objects.

Case Study	$ L $	States	Transitions	Model Const.(s)	Synthesis (s)
1:1	7	8,480	42,240	0.161	0.515
	9	43,848	298,080	0.496	2.679
	11	148,608	1,288,704	1.671	11.522
	13	393,800	4,166,400	5.216	21.126
Prob	7	9200	63440	0.148	1.002
	9	45,864	442,008	0.551	5.885
	11	152,928	1,905,120	1.862	23.822
	13	401,720	6,156,920	5.692	75.021

TABLE II: Abstraction & Synthesis comp. times for 5 objects.

$ L $	States	Transitions	Model Const. (s)	Synthesis (s)
4	148	387	0.052	0.037
5	4,896	17,184	0.248	0.688
6	43,416	190,107	1.101	7.366
7	217,600	1,144,320	4.406	38.768
8	787,500	4,846,875	5.944	60.933
9	2,304,288	16,280,352	90.314	677.584

task completion. We see that the computation time grows exponentially as the state space increases for fixed  $|O|$  and varying  $|L|$  and vice versa.

We also benchmark scenarios where there is 5% chance of human termination at every state. In contrast to [3], [7], where a parameter  $k$  was used to constrain the number of human interventions, this approach allows greater flexibility and a more intuitive model while still allowing the robot to win the game. The computation times are shown in Fig. 5c and Fig. 5f. Similar to the previous scenario, computation times grow exponentially as the state space increases. For both scenarios, the model construction time, while increasing, is relatively small compared to the synthesis time.

For all of the experiments using our modified implementation, PRISM-games required at most 8 GB of RAM. Table I reports the size of the game and time for model construction and strategy synthesis. Table II illustrates how the abstraction grows for the 1:1 scenario. The Python script runs out of memory for 5 objects and 10 locations for both scenarios.

**Physical Experiment: Tic-Tac-Toe with “Trembling Hand”.** Recall the game of tic-tac-toe in Fig. 1, where human and robot players alternate turns placing markers in empty cells. Tic-tac-toe can be solved using min-max, but

only under the assumption that the robot does not fail to complete an action, i.e., reactive synthesis cannot capture stochasticity in the robot’s ability to correctly place a marker at its desired location as per the strategy. While an MDP can capture the stochastic outcomes, it cannot model the strategic nature of the human. On the other hand, using the stochastic games model, we can account for the strategic and stochastic nature of both players. Fig. 6 illustrates a run of the game.<sup>1</sup>

In our experiment, we have stochasticity in placing the marker for both the robot and the human. We model this probability of marker placements (say a normal distribution with standard deviation of 1-cell width) as the uncertainty in the player’s action. We restrict both the robot and human to not be able to place the marker outside the cells or in an already occupied cell. Stochastic games allows us to reason over possible human and robot failure. Depending on the task, the robot can move so as to either maximize its chances of winning the game or the chances of human failure. In both case studies, the robot starts the game. We specify these tasks as  $P_{\max=?}[F(\text{“RobotWin”})]$  and  $P_{\min=?}[F(\text{“HumanWin”})]$ . The emergent behavior for the robot under specification 1 is to initially place its marker in the middle. This maximizes its chances of winning while reducing the number of unoccupied cells, which reduces the probability of failure in future robot actions. As the game progresses, we observe that the robot places its marker near crowded cells with fewer empty cells around it. For the second specification, we observe that the optimal action, initially, for the robot is to place its marker in the middle. Next, the robot places markers to maximize its winning probability while ensuring as many empty locations as possible for the next optimal human action.

## VI. CONCLUSION

We present a framework for robot manipulation based on stochastic games. Stochastic games subsume the expressivity of reactive and probabilistic synthesis proposed in previous works. We illustrate the efficacy of our approach through various scenarios and discuss the emergent behavior. Future work should examine symbolic approach to scale to more objects and locations. Additional work can examine modeling of uncertain observations, reasoning over other agents strategies, concurrent games or games with varying rewards.

<sup>1</sup>Video of more runs: <https://youtu.be/UUBW7QEw6Ng>



## REFERENCES

- [1] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *Int. Conf. on Robotics and Automation*. Rome, Italy: IEEE, 2007, pp. 3116–3121.
- [2] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 211–236, 2018.
- [3] K. He, A. M. Wells, L. E. Kavraki, and M. Y. Vardi, "Efficient symbolic reactive synthesis for finite-horizon tasks," in *2019 Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8993–8999.
- [4] S. Junges, N. Jansen, J.-P. Katoen, and U. Topcu, "Probabilistic model checking for complex cognitive tasks—a case study in human-robot interaction," *arXiv preprint arXiv:1610.09409*, 2016.
- [5] A. M. Wells, Z. Kingston, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Finite horizon synthesis for probabilistic manipulation domains," in *Intl. Conf. on Robotics and Automation*. IEEE, 2021.
- [6] A. Abate, J. Gutierrez, L. Hammond, P. Harrenstein, M. Kwiatkowska, M. Najib, G. Perelli, T. Steeples, and M. Wooldridge, "Rational verification: game-theoretic verification of multi-agent systems," *Applied Intelligence*, vol. 51, pp. 6569–6584, 2021.
- [7] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Reactive synthesis for finite tasks under resource constraints," in *Int. Conf. on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada: IEEE, 2017, pp. 5326–5332.
- [8] M. Kwiatkowska, G. Norman, D. Parker, and G. Santos, "PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time," in *Proc. 32nd International Conference on Computer Aided Verification (CAV'20)*, ser. LNCS, vol. 12225. Springer, 2020, pp. 475–487.
- [9] A. M. Wells, "Stochastic games for robotics." [Online]. Available: [https://github.com/andrewmw94/stochastic\\_games\\_for\\_robotics\\_code](https://github.com/andrewmw94/stochastic_games_for_robotics_code)
- [10] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.
- [11] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, vol. 13, 2013, pp. 854–860.
- [12] S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi, "Symbolic LTLf synthesis," in *Proc. of the 26th Intl. Joint Conf. on Artificial Intelligence*. AAAI Press, 2017, pp. 1362–1369.
- [13] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.
- [14] C. I. Vasile and C. Belta, "Reactive sampling-based temporal logic path planning," in *Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4310–4315.
- [15] E. M. Wolff, U. Topcu, and R. M. Murray, "Efficient reactive controller synthesis for a fragment of linear temporal logic," in *Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 5033–5040.
- [16] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Automated abstraction of manipulation domains for cost-based reactive synthesis," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 285–292, 2018.
- [17] K. Muvvala, P. Amorese, and M. Lahijanian, "Let's collaborate: Regret-based reactive synthesis for robotic manipulation," in *Intl. Conf. on Robotics and Automation*. IEEE, 2022, pp. 4340–4346.
- [18] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd Intl. Conf. on Computer Aided Verification (CAV'11)*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.
- [19] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.
- [20] F. Miao, Q. Zhu, M. Pajic, and G. J. Pappas, "A hybrid stochastic game for secure control of cyber-physical systems," *Automatica*, vol. 93, pp. 55–63, 2018.
- [21] L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu, "Controller synthesis for autonomous systems interacting with human operators," in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ser. ICCPS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 70–79.
- [22] A. M. Wells, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "LTLf synthesis on probabilistic systems (online version)."
- [23] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Model-free reinforcement learning for stochastic games with linear temporal logic objectives," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 10 649–10 655.
- [24] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, R. Brachman, F. Rossi, and P. Stone, *An introduction to the planning domain definition language*. Springer, 2019, vol. 13.
- [25] G. De Giacomo and M. Y. Vardi, "Synthesis for LTL and LDL on finite traces," in *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, vol. 15, 2015, pp. 1558–1564.
- [26] M. Kwiatkowska, G. Norman, D. Parker, and G. Santos, "Automatic verification of concurrent stochastic systems," *Formal Methods in System Design*, vol. 58, no. 1-2, pp. 188–250, 2021.
- [27] A. Condon, "The complexity of stochastic games," *Information and Computation*, vol. 96, no. 2, pp. 203–224, 1992.
- [28] K. Muvvala and M. Lahijanian, "Efficient symbolic approaches for quantitative reactive synthesis with finite tasks," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 8666–8672.
- [29] A. Condon, "On algorithms for simple stochastic games," in *Advances in Computational Complexity Theory, volume 13 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1993, pp. 51–73.
- [30] K. Chatterjee and T. A. Henzinger, *Value Iteration*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 107–138.