bayesnet

1 Introduction

Much of human social life occurs in contexts where people must coordinate their actions with those of others. From party planning to flying a rocket ship to the moon, groups of people have accomplished great things by reasoning as a team and engaging in jointly intentional behavior [**?**]. Indeed, some have argued that, because most other animals lack the capacity to work adaptively as cohesive unit across many domains, team reasoning may be the hallmark of human sociality [**?**].

Artificial agents are becoming ubiquitious in everyday life. For these agents to collaborate effectively with humans, however, they will need behave in human-like ways. Unfortunately for agent designers, there is no simple behavioral rule (or set of rules) that could be prescribed to cover all contexts, communities, and cultures. People perpetually learn new rules and alter their presumed roles,[1] and so must artificial agents.

Most attempts to build collaborative, artificial agents begin with a model (however flawed) of human behavior. One possibility is to assume we are *Homo economicus*—a term from the 1800s that portrays individuals as being guided by self interest, and taking actions without regard to their impact on the collective. Alternatively, one might assume an individual to be *Homo sociologicus*—a more recent term depicting individuals as pursuing goals imprinted on them by society.

The homo-sociologicus model was not intended as a complete description of human behavior, which is clearly self interested at times. But, psychologists and behavioral economists have also more or less rejected the homo-economicus model [**?**, **?**], proposing countless homo-sociologicus-leaning models in its stead [**?**]. Consistent with these findings, this proposal is founded on the assumption that **social preferences** hold the key to successful collaborations. Social preferences are in part **distributive**, meaning they evaluate how resources are allocated, e.g., efficiently or equitably; and they are in part **reciprocal**, meaning they evaluate the perceived intentions of the actors in the environment.

We plan to model interactions among such social agents, both human and artificial, in the framework of game theory. Following Bacharach [**?**], not only do we assume that preferences are social rather than individual, we go one step further and assume that decision making takes the form of "What should *we* do?" rather than "What should *I* do?" In other words, we move from the usual assumption of best-reply reasoning [**?**] in games to one of **team reasoning** [**?**].

We call optimal decision making, when agents hold social preferences and employ team reasoning, **socially rational** behavior. Social rationality is a blend of the *Homo economicus* and *Homo sociologicus* concepts: agents optimize a **social utility function** (i.e., a representation of social preferences), which is sufficiently rich to incorporate perceived (possibly inferred, possibly imagined) societal benefits.

We contend that the computational process by which collaborative behavior emerges strongly depends on social rationality, coupled with learning via repeated interaction. Specifically, the computational process is thus: 1. agents represent social preferences using a social utility function tailored to the environment and the actors;[2] 2. based on their estimate of this function, agents use team reasoning to make plans to act; 3. agents act, simultaneously or sequentially, as is appropriate in their environment; 4. agents update their estimates of their social utility functions. Steps 2 through 4 repeat so long as the interaction continues.

When agents make a plan that optimizes a social utility function, they are choosing actions that jointly optimize the behavior of the collective. When agents update their estimate of the social utility function, they do so using the observed history of joint actions in the environment. If this process stabilizes, the social utility function represents the preferences of the community as a whole, and the equilibrium implements a joint plan of action that optimizes that social utility function. Indeed, in our computational model, emergent collaborative behavior and social preferences are expected to reinforce one another.

Classical game theory assumes that utility functions are exogenously determined: i.e., agents come to the table with fixed extrinsically-set utilities. However, there is broad agreement in the judgment and decision-making community that preferences are *constructed*, not elicited [**?**], in response to available choices. Likewise, our computational framework does not assume that social preferences exist extrinsically; rather,

---

[1]As Shakespeare said, "one man in his time plays many parts."

[2]This utility function is described in terms of features of the environment and the actors; as such, it is applicable to other environments and actors with similar features.

they are constructed in response to observed behaviors. In our conception of social rationality, agents *learn* social preferences through repeated play of the game.

A social utility function is an immediate generalization of the usual notion of an individual's utility function, but it ascribes value to the outcome as it relates to all actors in the environment. As such, it can be used to capture important elements of an artificial agent's decision making, including (1) how it values the utility of others [?], (2) how it values the utility of intermediate goals [?], and (3) how it values sanctions against others who exhibit anti-social behavior [?].

We further assume that social utilities can be broken down into two components—an objective component, which is usually a direct function of the rules of the game, and a subjective component, which captures notions of distributivity and reciprocity. Often, the rules of a game alone give rise to multiple (objective) equilibria, so that agents face a difficult equilibrium-selection problem (e.g., [?]) in the underlying game. We contend that socially rational behavior, in which agents optimize a learned social utility function that reflects (constructed) social preferences, can provide a solution to this challenging, and often elusive, problem.

Under the assumption that human agents are, perhaps boundedly but nonetheless ideally, socially rational creatures, we propose to design and build socially-rational artificial agents that learn through repeated play, with the aim being for such agents to collaborate effectively with humans.

In this proposal, we set out to demonstrate our potential for success, by showing that our computational model is viable. To this end, we present a preliminary set of simulation results showing how reinforcement-learning agents, interacting repeatedly in a grid-game environment, can represent social preferences using a social utility function, so that eventually their emergent collaborative behavior and their discovered social preferences reinforce one another. We then go one step further and show that our approach can also learn (in an off-line, batch fashion) from human–human trace data so that the collaborative behavior that is learned mimics that which is expressed in the trace data. While these two experiments (on machine–machine, interactive and human–human batch data) are indeed promising, it remains to extend these techniques to human–machine interactions, preferably in experimental worlds with embodied robots. **Amy: HUMAN TRACE DATA!**

The ultimate goal in developing artificial agents that act intelligently in multi-agent scenarios is to apply them to real-world problems. To a limited extent, this goal has already been achieved: artificial agents trade stocks and bid in online ad auctions. In these two market environments, rationality may well be an appropriate model of behavior for the participating agents. However, an artificial agent that plans its collaboration by assuming the humans in its environment will act selfishly and/or are driven by extrinsic utility functions is unlikely to be successful in its collaborations [?, ?]. Instead of individual rationality as the underlying principle guiding human behavior, this proposal is grounded in the assumption that humans are socially rational beings whose preferences and behaviors emerge and reinforce one another through repeated interactions. For artificial agents to successfully collaborate with humans, they should be thus, too!

2   Grid Games

A grid game is a game played by two agents on a grid, in which each agent has a goal. See, for example, Figure 1a, which is a 3x5 grid in which the two agents' initial positions are one another's goals: Orange begins in position (1,2), Blue's goal; and Blue begins in position (5,2), Orange's goal. We refer to grid positions using x-y coordinates, with $(1, 1)$ as the bottom left position.

One grid game proceeds in rounds, each round consisting of multiple turns. On each turn, the agents choose one of five actions (N, S, E, W, or wait), which are then executed simultaneously. In the most basic setup, agents transition deterministically, and there is no tie-breaking when two agents collide.[3] Instead, if their chosen actions would result in a collision with one another, neither agent moves. A round ends when either (or both) players move into their goal, or when a maximum number of turns has been taken.

We present five two-agent grid games, specifically designed to vary the level of coordination required, while at the same time allowing agents to defend against uncooperative partners. In these games, one

---

[3]It is a simple matter to vary these rules within our infrastructure, as future experimental design might dictate.

approach is for an agent to cooperate blindly with its opponent by simply moving out of the opponent's way, and hoping the opponent then waits for the agent to catch up. However, such strategies can be exploited by uncooperative ones that proceed directly to the goal as soon as their path is unobstructed.

To distinguish "unsafe" from "safe" cooperation, we devised a new categorization for strategies in our grid games. We call strategies that allow for cooperation, while at the same time maintain a defensive position in the event that the other agent is uncooperative, **cooperative defensive** strategies. More formally, an agent's strategy is **cooperative** (C) if it is one that allows both it and its opponent to reach their goals, while an agent's strategy is **defensive** (D) if its opponent does not have a strategy that allows it to reach its goal strictly first. A cooperative defensive (CD) strategy is both cooperative and defensive.

We now proceed to describe a sample set of grid games, and equilibria comprised of CD strategies (when they exist), to illustrate the kinds of interactions we plan to study. Our first example, Hallway, is depicted in Figure 1a. This game is one in which the agents can choose to coordinate, for example, if both agents agree upon a joint strategy where one agent moves along the top row and the other along the bottom, without interfering with one another. But, an agent could choose to "defect" from this joint strategy, by proceeding straight to its goal. There are CD strategies, however, which defend against such non-cooperative behavior.

For example, if Orange moves south initially to $(1, 3)$ and Blue moves west to $(4, 2)$, Orange might choose to return and remain on its goal until Blue retreats to $(4, 3)$ or $(4, 1)$, at which point the players are equidistant from their goals, and both can reach them safely. This joint strategy is an equilibrium comprised of CD strategies, since Orange and Blue both remain in positions where they have the ability to block their opponents until they both have unobstructed equidistant paths to their respective goals.

The grid in Figure 1b (Intersection) requires Blue to defend against the possibility of Orange behaving uncooperatively, which it can achieve by squatting on the orange goal. Orange can then move to $(3, 1)$ where both agents are equidistant from their goals. Therefore, this game also has an equilibrium comprised of CD strategies for both players.

This equilibrium is not the shortest path, however. Purely cooperative agents in this game could adopt a joint strategy in which Blue moves east, while Orange waits a single step, before both agents proceed into their goals. This strategy profile is not defensive for Blue though, because it does not have the opportunity to observe if Orange will cooperate (wait) or defect (go north), and therefore cannot defend itself if Orange decides to head straight toward its goal.

Figure 1c (Door) is a grid that requires coordination to navigate through the narrow center space at $(3, 2)$. Any equilibrium comprised of CD strategies for this grid must be asymmetric, because it requires one agent to cede to the other agent the center cell. For example, if Orange chooses to cede that cell, it should step west into $(2, 3)$ while Blue steps south into $(3, 2)$. Then, Orange needs to step east back into $(3, 3)$ to prevent Blue from marching straight into its goal. Only when Blue agrees to step aside to $(2, 2)$ will they both be equidistant from their respective goals and in position to cooperate. This intricate pattern of first giving way to the opponent, and then forcing them to step around later represents an equilibrium comprised of CD strategies, since both agents are able to prohibit their opponent from reaching the goal first, but still leaves open the possibility for them both to reach their goals, cooperatively.

In the grid in Figure 1d (Long hall), Blue begins one step closer to its goal than Orange does. However, Orange can squat on the blue goal until Blue chooses to cooperate by taking one step back. If Orange can predict when Blue steps back, then Orange can take one step closer to its goal while Blue steps further away, in which case only Orange would reach its goal. The strategy that minimizes the risk to either agent requires that Blue wait one turn initially, while Orange moves toward its goal. These two strategies comprise a CD equilibrium.

Our last grid, shown in Figure 1e (No compromise), requires not only cooperation, but both agents must also exhibit trust for one another, or both agents cannot arrive at their goals at the same time. For example, Orange may sit on Blue's goal so that Blue can move to $(1, 2)$. Then, Blue must wait two turns before both agents are equidistant from the goals. If Blue defects and moves south into $(1, 1)$ while Orange moves south into $(2, 2)$, Orange still has the opportunity to go back up north to block Blue from reaching its goal. However, if Blue moves south into $(1, 1)$ when Orange steps east into $(3, 2)$, Blue will arrive at its goal sooner. Therefore, a trust spanning multiple rounds is required for the agents to effectively cooperate in this game.
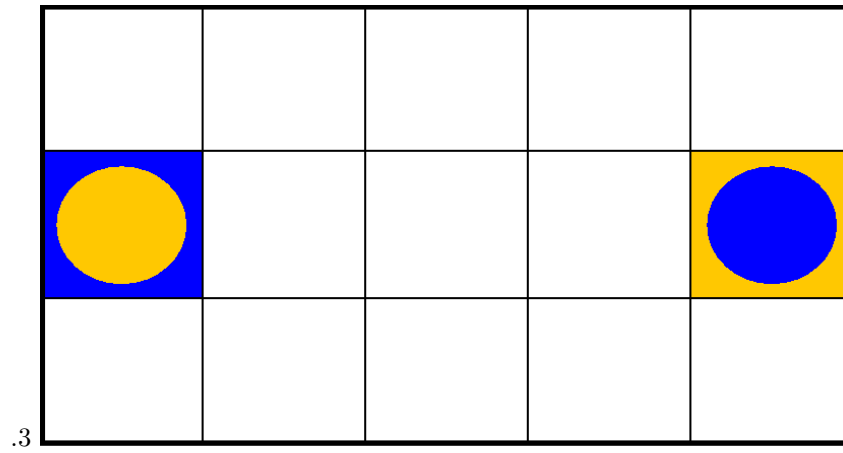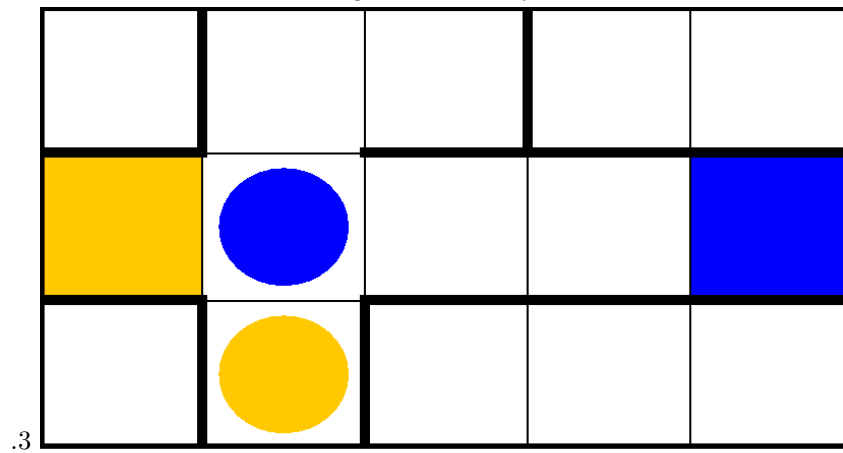
Figure 1: Hallway
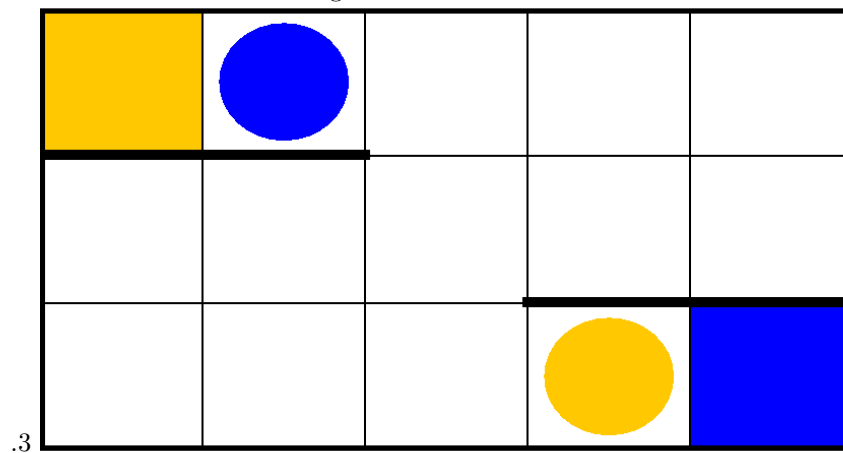

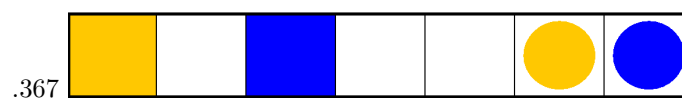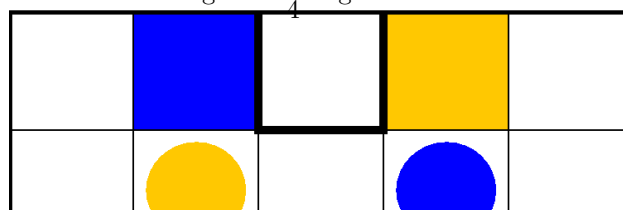Figure 2: Intersection

Figure 3: Door

Figure 4: Long Hall

No equilibrium in CD strategies exists for No Compromise. The game is like Door in that only one player can go move through the middle cell at a time. Unlike Door, however, it is not possible for the agents to simultaneously maintain a defensive position and to signal cooperation, because any cooperative move leads to an asymmetric situation in which the agents are no longer equidistant from their goals. As a result, after one agent cooperates, there is always an incentive for the other agent to defect, and there is nothing the cooperative agent can do to defend itself. Note however, that if Orange sits on the blue goal while Blue walks to $(1,1)$ and then Blue cooperates by waiting for Orange to walk to $(1,3)$, Blue's policy is CD. Still, Orange cannot respond in kind with a CD strategy; the aforementioned strategy is C.

Taking these five grid games as an initial testbed, we performed two pilot studies: the first involved simulations of artificial agents playing against one another; the second pitted humans against other humans on Mechanical Turk []. **Amy: add citation to workshop paper** We report on the results of these preliminary studies next.

Simulation Experiments We carried out a set of simulation experiments with Q-learning in the grid games presented. We denote the outcome of a round using a pair of letters, where **G** means the agent reached the goal and **N** means the agent did not reach the goal. The first letter in the pair represents the agent's own outcome and the second represents its opponent's outcome. For example, **GN** is used to denote that the agent reaches its goal while its opponent does not.

As two Q-learning algorithms are not guaranteed to converge in self-play, we arbitrarily stopped the learning after 30,000 rounds, and checked the strategies learned. In spite of Q-learning not explicitly seeking outcomes with high social welfare, it very reliably identified cooperative strategies leading to **GG** outcomes.

In a second set of Q-learning experiments, we examined the impact of endowing agents with **social preferences**. That is, their utilities no longer depend solely on their own successes and failures—they depend on those of other agents in the environment as well.

To make this idea more precise, we draw a distinction between **objective** and **subjective** reward.[4] Objective reward is the usual reward signal provided to the agent from the environment, and by which behavior is judged. Standard reinforcement-learning agents, such as Q-learners, seek to optimize their own objective reward signal—we call this preference the **selfish** preference because these agents are only concerned with their own outcomes.

In some environments, however, it useful to distinguish this objective reward from subjective reward—the perceived quantity the agent is optimizing. Previous work has shown that optimizing subjective reward can sometimes lead agents to be more effective in their acquisition of objective reward [**?**]. (Indeed, we reach this same conclusion in our experiments.)

Considering the four different outcomes in these games—**GG**, **GN**, **NG**, **NN**—there are 75 different possible preference orderings (allowing for ties). The selfish ordering that ignores the opponent's outcome is one of these: **GG** $\sim$ **GN** $\succ$ **NG** $\sim$ **NN**. Nine of the 75 orderings are consistent with the selfish ordering, strictly preferring **Gx** to **Nx** for all **x**.

Of particular interest is the **fair** preference, which we define as the objective reward of the agent minus 25% of the difference between its own and the opponent's objective rewards: $r_s = r_a - 0.25\,|r_a - r_o|$, where r_s is the agent's subjective reward, r_a is the agent's objective reward, and r_o is the opponent's objective reward.[5] By incorporating this fairness term into the agent's subjective reward, it strictly prefers the following ordering: **GG** $\succ$ **GN** $\succ$ **NN** $\succ$ **NG**. That is, the agent prefers making it to its goal as opposed to not, but it additionally prefers that the opponent only get to its goal if the agent itself does as well. To say it another way, a fair agent wants its opponent to win with it or lose with it.

Figure 4 shows the result of the selfish and fair agents playing against others of the same type in each of our test grid games. Of the nine orderings, only three (all variations of the fair preference ordering) achieve consistent cooperation in self play across all five grid games. Consequently, fair agents obtain higher total (objective) rewards than others. We also ran all nine preference orderings against
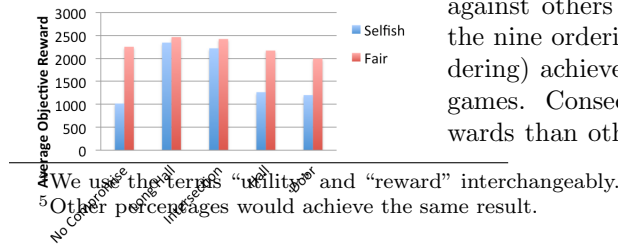
[4]We use the terms "utility" and "reward" interchangeably.
[5]Other percentages would achieve the same result.

Figure 7: Average score of Q-learners in self play (30,000 rounds)

5

one another. The average scores (across both players) in games involving a fair agent tend to be higher than the average scores in games not involving a fair agent. We also analyzed the types of strategies learned by fair and selfish Q-learners after multiple simulations of various configurations. Interestingly, we found that Q-learners with fair preferences tend to find CD strategies more often, especially when paired with selfish agents.

In summary, Q-learners naturally learn to cooperate in the grid games studied, discovering equilibria comprised of CD strategies when they exist. Moreover, cooperation can be induced in these games by manipulating the Q-learners' subjective rewards to incorporate the success of others.

Turk Experiments We also conducted analogous experiments with humans playing grid games on Mechanical Turk. The results involving human subjects playing grid games are consistent with the results obtained in our simulations of Q-learners playing grid games. In both sets of experiments, more cooperation was achieved when the treatment incorporated social preferences.

While the subjective utility functions of artificial agents are within our control, so that we can perhaps lead artificial agents towards collaborative behavior, the subjective utility functions of humans are not. Nonetheless, behavioral economists often infer approximations of utility functions from experimental data by assuming some underlying structural form, and then estimating the relevant parameters [**?**, **?**].****Amy: CHECK REFERENCES!**** Likewise, one of the primary intended uses of our Turk experimental framework is to generate trace data of humans playing grid games and learning collaborative behavior (such as trust, CD, etc.), so that we can then proceed to infer utility functions and relate them back to the behaviors they produce.

3   Computational Framework

Recall, from them Introduction, our computational model: 1. agents represent social preferences using a social utility function; 2. based on their estimate of this function, agents use team reasoning to make plans to act; 3. agents act, simultaneously or sequentially, as is appropriate in their environment; 4. agents update their estimates of their social utility functions.

While certain aspects of social preferences (e.g., fairness) are broadly applicable, Step 1 generally requires a representation that is tailored to the specific environment and actors within. Likewise, Step 3 involves a straightforward simulation in grid games, but is otherwise specific to the robots and their environment.

Only Steps 2 and 4 are sufficiently generic to apply across environments. Step 2 is a **planning** step; as planning is a well-defined optimization problem, it can be accomplished using any off-the-shelf planner for stochastic sequential decision making environments [**?**, **?**, **?**, **?**, **?**, **?**]. Step 4, on the other hand, is not a solved problem; indeed, in this proposal, we describe an innovative approach to this problem.

Step 4 can be thought of as a **learning from demonstration** problem [**?**], in which an agent learns how to behave in an environment by observing an expert. That is, an agent is presented with a data set consisting of multiple examples of behaviors (e.g., state–action trajectories through a Markov decision process), and is tasked with the objective of learning a policy capable of (closely) reproducing the data.

Two widely studied approaches to this problem include **imitation** and **intention** learning. Imitation learning is a supervised learning approach in which an agent learns a mapping from states to actions, with the expert's behavior serving as training data [**?**]. After training, a learner follows its learned policy directly.

Learning intentions [**?**], in contrast, is often framed as an **inverse reinforcement learning** (IRL) problem [**?**, **?**]. Here, the agent's goal is to learn rewards that motivate the expert's behavior, after which it applies a planning algorithm to derive a policy that is consistent with those learned rewards. A strength of intention learning is that even simple reward functions can capture complex behavior, leading to greater generalization capabilities (i.e., appropriate behavior in as-yet-unforeseen states) than the more direct approach taken by imitation learning.

We are proposing to use IRL in Step 4. Preliminary experiments, described herein, with one off-the-shelf IRL algorithm, demonstrate the potential of our computational model to facilitate collaborative behavior among artificial agents. But to better support our goal of human–machine collaboration, we also propose a new approach to IRL in which demonstrations are not, by default, assumed to have been generated by an expert. With this additional power, our IRL approach can be applied to all past interactions among agents, separating the successful from the unsuccessful. Indeed, in the computational process we put forth, learning is online, not off, so not all demonstrations should be interpreted as successful. On the other hand, as past interactions are shared context across agents, ideally they could *all* provide crucial guidance for identifying cooperative behavior, and that is what we set out to achieve.

3.1 Mathematical Models

As grid games are examples of stochastic games [**?**, **?**], we use this game-theoretic framework to model agent interactions. Like econometricians [], we assume a structural form for utility functions; and because we are interested in building socially rational artificial agents, we endow our agents with a (potentially) social utility function. We then develop a batch learning algorithm by which an artificial agent can, by learning from demonstrations, recover a social utility function that supports exactly one of many equilibria (assuming the demonstrations exhibit exactly one of many equilibria). We also develop an interactive learning algorithm by which two agents can simultaneously learn a social utility function, and converge to one among a plethora of equilibria, thereby endogenously solving the equilibrium selection process.

Markov decision processes   A **Markov Decision Process** (MDP) is a model of a single-agent decision making problem, defined by the tuple (S, A, T, R), where S is the set of states in the world; A is the set of actions that the agent can take; $T(s' \mid s, a)$ defines the transition dynamics: the probability the environment transitions to state $s' \in S$ after the agent takes action $a \in A$ in state $s \in S$; and $R(s, a, s')$ is the reward function, which returns the reward the agent receives when environment transitions to state s' after the agent takes action a in state s.

The goal of planning or learning in an MDP is to find a policy $\pi : S \rightarrow A$ (a mapping from states to actions) that maximizes the expected future discounted reward under that policy: $E^\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right]$, where $\gamma \in [0, 1]$ is a discount factor specifying how much immediate rewards are favored over distant rewards.

To find an optimal policy, many algorithms compute the optimal state ($V^*(s)$) and state-action ($Q^*(s, a)$) value functions that specify the expected future discount reward under the optimal policy from each state, and from taking an action in a state and then following the optimal policy respectively. Given these functions, the optimal policy is derived by taking an action in each state with the maximum Q-value: $\pi(s) \in \arg\max_{a \in A} Q(s, a)$ [**?**].

Inverse reinforcement learning   Although there are multiple IRL formalizations and approaches, they all take as input an MDP together with a **family** of a reward functions R_Θ that is defined by some parameter space Θ, and a dataset D of trajectories (where a trajectory is a finite sequence of state-action pairs: $\langle (s_1, a_1), ..., (s_n, a_n) \rangle$). The algorithms then output a specific parameterized reward function $R_\theta \in R_\Theta$, which induces a policy that is consistent with the input trajectories. Different IRL algorithms frame the objective function for policy consistency differently. One common approach is to treat the search problem as a probabilistic inference problem [**?**, **?**, **?**, **?**].

For example, in the maximum-likelihood setting [**?**], the goal is to find a reward function parameterization that maximizes a the likelihood of the data:

$$\theta \in \arg\max_\theta L(D \mid R_\theta) = \prod_{t \in D} \prod_{i}^{|t|} \pi_\theta(s_i, a_i), \tag{1}$$

where $\pi_\theta(s, a)$ is a stochastic policy defining the probability of taking action a in state s when the parameterized reward function to be maximized is R_θ. Typically, the Boltzmann (softmax) stochastic policy over the Q-values is used: $\pi_\theta(s, a) = \frac{e^{\beta Q_\theta(s, a)}}{\sum_{a' \in A} e^{\beta Q_\theta(s, a')}}$, where $Q_\theta(s, a)$ is the Q-function when the reward function

is parameterized by θ. Popular methods for solving Equation ?? (i.e., for carrying out maximum-likelihood IRL (MLIRL) include gradient ascent and expectation-maximization.**Amy: do these algorithms need references?**

In Bayesian IRL, the goal is to compute a posterior distribution over reward function parameterizations:

$$\Pr(R_\theta \mid D) \propto \Pr(D \mid R_\theta)\Pr(R_\theta) \tag{2}$$

Bayesian IRL algoriths typically rely on Markov chain Monte Carlo methods. **Amy: add reference** More recently, work has also commenced on nonparametric Bayesian IRL methods [].

IRL algorithms are typically computationally demanding because they require planning in their inner loops. Compatible with all of the above algorithms, **receding horizon IRL** (RHIRL) [**?**] addresses this limitation by replacing the usual infinite-horizon policy with a receding horizon controller (RHC) that only plans out to some finite horizon from any given state, thereby bounding computation time. An RHC tends to steer an IRL algorithm towards a **shaping reward function** [], short-term rewards that guide the agent without it having to plan too far ahead. Consequently, RHIRL allows us to plan with a short horizon, thereby saving on computation time.

Stochastic games The stochastic games formalism can be viewed as an extension of MDPs to the multi-agent case [**?**]. A **stochastic game** is defined by the tuple (I, S, A^I, T, R^I), where I is an index set of agents in the environment; S is the set of states of the environment; A^I is set of actions for each of the agents with A^i denoting the action set for agent $i \in I$; $T(s' \mid s, j)$ is the transition dynamics specifying the probability of the environment transitioning to state $s' \in S$ when the *joint action* $j \in \times_i A^i$ of all agents is taken in state $s \in S$; and R^I is a set of reward functions for each agent with $R^i(s, j, s')$ denoting the the reward received by agent $i \in I$ when the environment transitions to state $s' \in S$ after the agents take joint action $j \in \times_i A^i$ in state $s \in S$.

The goal in a stochastic game is to find a joint strategy that satisfies some solution concept. Different solution concepts for stochastic games have been explored in the past including minimax, Nash, correlated, and CoCo equilibria [**?**, **?**, **?**, **?**]. There are problems with these approaches, however. First, the resulting planners must solve for game-theoretic equilibria in an inner loop, a problem which in the case of Nash equilibrium, for example, is notorious for its computational intractability [**?**]. Second, in the general case of non-constant sum games, none of the planners that make reasonable assumptions about agent behavior yield unique joint plans, and none has solved the ensuing equilibrium selection problem suitably.

Immediately generalizing from the case of MDPs, the goal of inverse reinforcement learning in stochastic games is to learn a set of reward functions for the stochastic game that describe an environment that would motivate the agents to behave in a way that is consistent with the observed behavior under some solution concept such as a Nash equilibrium [**?**]. This problem, however, is exceedingly difficult, because the planning that is necessary in the inner loop of an IRL algorithm is subject to the challenges identified by the equilibrium planners mentioned above. In this project, we plan to develop IRL algorithms that facilitate equilibrium selection in a stochastic game.

Social Reward Functions In our preliminary studies, we assume players' individual (game) rewards are known, and our goal is to recover a **social reward function** that combines these known rewards in such a way as to capture social preferences expressed in the joint play of the agents (when one exists).

Like the players' reward functions, a social reward function operates on states, joint actions, and next states: $R^S : S \times_i A^i \times S$. In our initial model, we assume the social reward function can be written as linear combination of a **team function**, which represents team goals, and a family of what we call **bias functions** (B_Θ) defined by some parameter space Θ. The team function takes as input a multi-agent reward function (R^I), and returns a single numeric "team" value for any state-joint action-next state triple. One example of such a function is total welfare (i.e., the sum of all agent rewards): $\mathcal{T}(R^I, s, j, s') = \sum_i R^i(s, j, s')$. The bias function family is similar in nature to a reward function family that would be input to a classic IRL algorithm, but operates on state-joint action-next state triples, thereby encoding a bias that motivates collaborative behavior in games. In addition to the parameters of the bias function, the social reward function may also include a parameter to trade-off the team rewards against the biases.

Algorithm 1 Batch_Learning($(I, S, A^I, T, R^I), \gamma, D, \mathcal{T}, B_\Theta$)

---

stochastic game (I, S, A^I, T, R^I); discount factor γ; multi-agent demonstrations D; team function $\mathcal{T}$; and parameterized bias function family B_Θ. $A^M := \times_i A^I$ Joint action set $R^M(s, a, s') := \mathcal{T}(R^I, s, a, s')$ Team reward function $R_\Theta^S(s, a, s') := R^M(s, a, s') + B_\Theta(s, a, s')$ Social reward function family $R_\theta^S := $ IRL$((S, A^M, T, R_\Theta^S), \gamma, D)$ R_θ^S Learned social reward function

---

To facilitate social reward functions that generalize, the bias function family must operate on a set of useful features. As part of this project, we plan to investigate whether existing state-of-the-art feature selection methods for RL [**?, ?, ?, ?**] work well in our setting, and then work towards developing new methods to the extent necessary.

3.2 Collaborative Learning Algorithms

Given our representation of social preferences by a social reward function that is parameterized by a family of bias functions, our approach to collaborative learning is simply to optimize those parameters to best match observed behavior. We consider two forms of collaborative learning. The first is **batch** learning, in which an agent learns offline from batches of demonstrations of other agents playing collaboratively. The second is **interactive** learning in which agents learn the social preferences of the collective via repeated interactions.

Batch Learning   The batch learning setting is similar to the standard IRL problem in that the algorithm is given a batch of demonstrations and a family of reward functions over which to search. In our case, however, the batch data consists of trajectories of states and joint actions in a stochastic game, and the reward function family is a social reward function family. To learn the parameterization of the social reward function family that best captures the behavior exhibited in the demonstrations, we convert the stochastic game into an MDP, and then use IRL to learn a parameterization of the reward function family in this MDP.

Pseudocode for the batch learning algorithm is shown in Algorithm 1. The algorithm takes as input a stochastic game; a discount factor; a set of multi-agent demonstrations (which perhaps adheres to some equilibrium: i.e., collaborative behavior); a team function; and a family of bias functions defined by some parameter space Θ. Since the demonstrations are a set of trajectories of state-joint action pairs, specifying the behavior of all the agents in the environment, the first step of the batch learning algorithm is to transform the stochastic game into an MDP. For the most part, this transformation is straightforward: the states are the same, the MDP action set is the space of joint actions, and the MDP transition dynamics still operate on joint actions (which is the action set in the MDP). A family of social reward functions is then defined as a linear combination of the input team function (which depends on the stochastic game's reward functions), and the family of bias functions.[6] After learning a social reward function, any single agent can then behave by selecting a joint action from the policy that results from it, and then selecting their individual action from the joint. That is, agent i takes action $j^i \in A^i$, where $j \in \arg\max_j Q_{R_\theta^S}(s, j)$.

Interactive Learning   In the interactive learning setting, an agent plays with a set of unknown agents, with whom it can potentially establish new collaborations. Pseudocode for our interactive learning algorithm is shown in Algorithm 2. The algorithm takes as input the same arguments as the batch learning algorithm, except it includes an index specifying which agent in the stochastic game the learner is, and it does not include the batch of demonstrations. The agent begins by initializing its policy arbitrarily.[7] The first round of interaction produces a trajectory of joint behavior that the learner can now learn from. It does so by running our batch learning algorithm on the single trajectory of experience acquired thus far. It then proceeds to follow the learned policy in the next round. After the second round completes, the agent now has two trajectories on which it can run batch learning to update its social reward function and its behavior. This process then repeats for all rounds of interaction. Note that updating the policy after each round

---

[6]A parameter controlling the linear combination of the team and bias function may be optimized as well.

[7]In practice, the policy can be initialized to the agent's role in the joint policy that follows from the team function alone.

Algorithm 2 Interactive_Learning$((I, S, A^I, T, R^I), k, \gamma, \mathcal{T}, B_\Theta)$

---

stochastic game (I, S, A^I, T, R^I); agent index k; discount factor γ; team function $\mathcal{T}$; and parameterized bias function family B_Θ. initialize policy π^k arbitrarily $A^M := \times_i A^I$ Joint action set $D := \{\}$ each round of play play round following π^k append new trajectory of play to D $R_\theta^S :=$ Batch_Learning$((I, S, A^I, T, R^I), \gamma, D, \mathcal{T}, B_\Theta)$ $\forall s \in S$, $\pi^k(s) := a^k \in A^k$ where $a \in \arg\max_a Q_{R_\theta^S}(s, a)$

---

requires computing the Q-values under the newly learned social reward function. However, in practice, this computation can be performed lazily for each state the agent observes in the next interaction.

Preliminary Results   Preliminary experiments suggest that both our batch and interactive collaborative learning algorithms can produce reasonable results in grid games. For batch learning, we hard code examples of agents following different equilibrium strategies, and show that our algorithm is able to recover the appropriate equilibrium when it is trained on demonstrations of behavior following that equilibrium. We also testing the agents on similar grid games after learning, to show that the learned behavior can generalize to other similar situations. For interactive learning, we show that agents can very quickly find an equilibrium, and that restarting learning from scratch can result in them finding a different one.

For both sets of experiments, we examine behavior in Hallway. The social reward function consists of a total welfare team reward function and a linear family of bias functions. The bias functions operate on 50 state-joint action binary features. The first 25 features are an indicator variable for selecting one of the 25 joint actions when the agents are in *conflict* and the latter 25 features are indicator functions for selecting one of the 25 joint actions when the agents are not a conflict. We define the agents to be in conflict when they are in the same row of the grid and each is closer to their opponent's goal than they are to their own. Additionally, when one of the agents is one step away from their goal, none of the features activate, thereby preventing premature termination on the joint task. Since these features directly encode social rewards for joint actions, rather than higher-level goals, when training we use RH-MLIRL with a horizon of 1. We do not expect this set of features to be complete or capable of capturing all the possible behaviors that might emerge in a grid game. So an important area for future work is to identify features that are both expressive and generalize well. Nevertheless, the features used here provide a proof of concept for our approach.

We tested our batch learning algorithm on datasets in which agents followed three different equilibrium strategies. The first (Figure 5a) shows the orange player going north and then traveling along the top row of the grid, while the blue player goes south and then travels along the bottom row. The second (Figure 5b) shows the blue player going north and then traveling along the top row while the orange player waits, then goes east, and then waits again for blue to catch up so they can enter their goals together. The third (Figure 5c) has the orange player go south and then traveling along the bottow row, while the blue player immediately goes west toward its goal and then waits two steps.

For each of these three equilibria we generated five demonstrations, two of which described the behavior exactly, and the other three of which contained slight deviations (for example, moving back to the center row before reaching the end). We then separately trained our batch algorithm on each set of demonstrations. In all cases, the learned social reward function motivated behavior that consistently replicated the data on which it was trained. Additionally, we planned using the learned social reward function in a larger 7x3 hallway grid game, and in all cases, obtained the corresponding equilibrium behavior.

In our interactive learning experiment, we played two interactive learning agents against one another. The agents played five matches, each one consisting of five rounds. Each match began fresh, without any knowledge of previous matches, to see whether different equilibria could emerge. The social reward function was initialized to the total welfare joint policy. In each match, the agents very quickly (typically after the first round) converged to an equilibrium that persisted for the rest of the match. However, in each match, a different equilibrium was learned, showing that agents employing our algorithm quickly adapt to the particular shared experience they have with other agents.

Figure 6 shows the results of learning in each of the five matches (with the top image showing the behavior in the first round and the bottom image showing the learned behavior). While different equilibria emerged
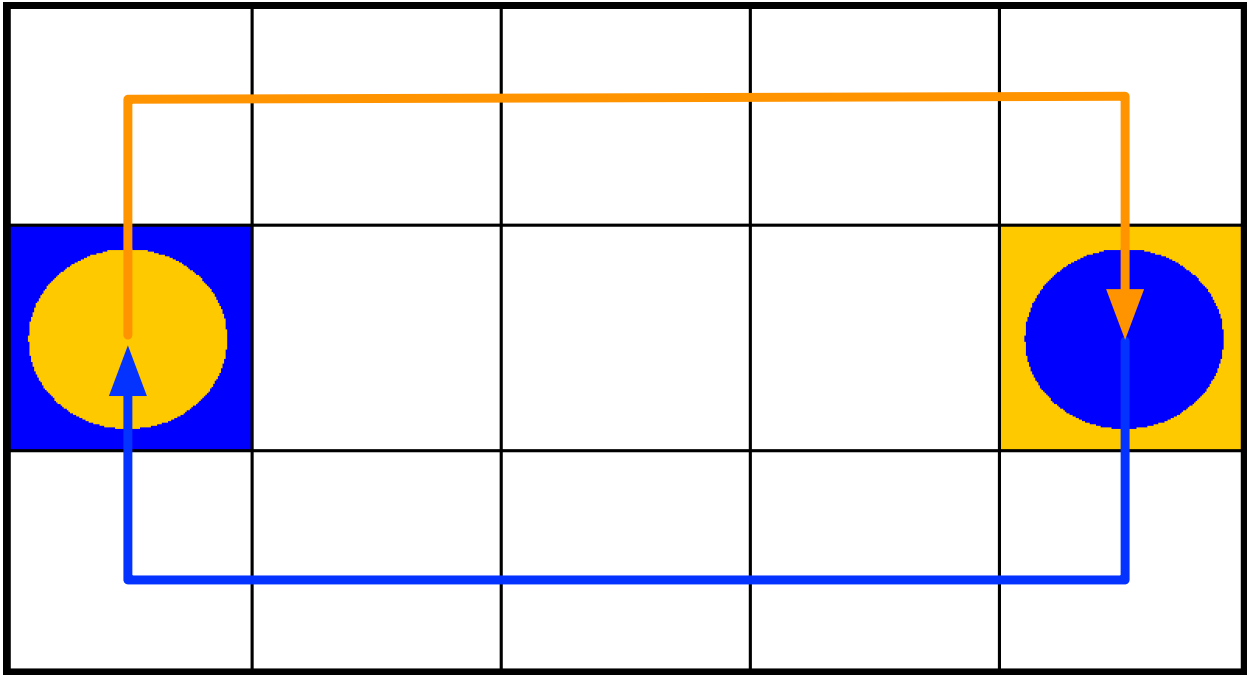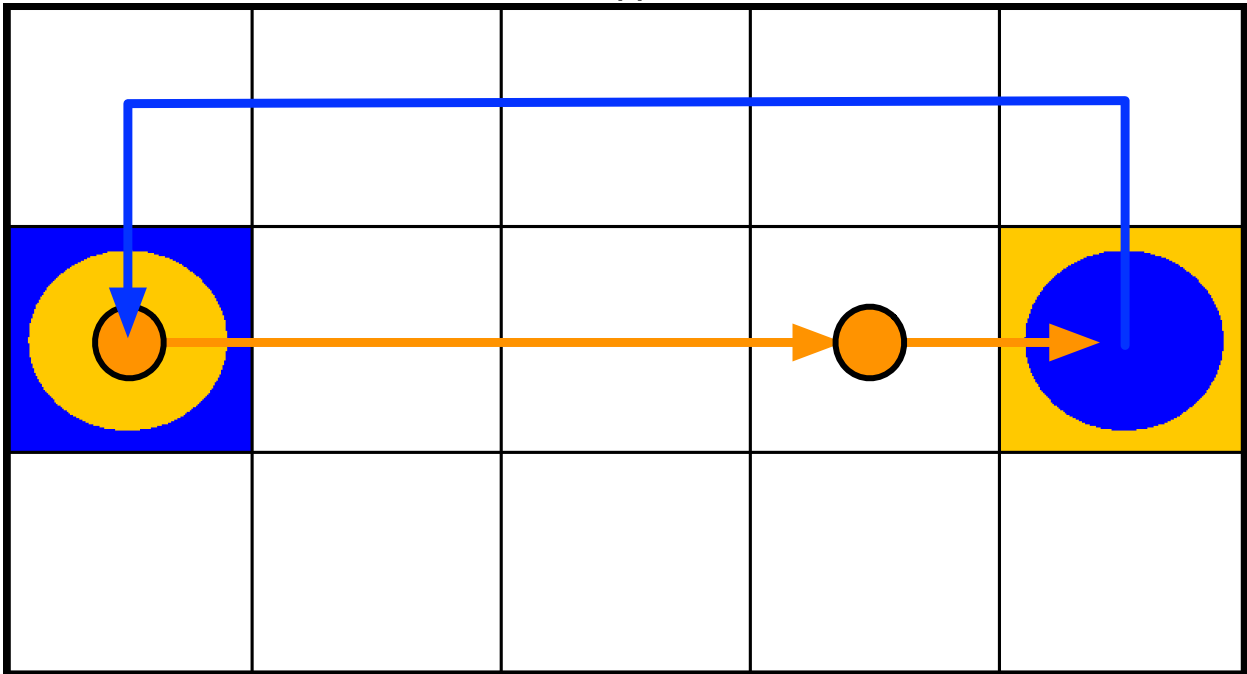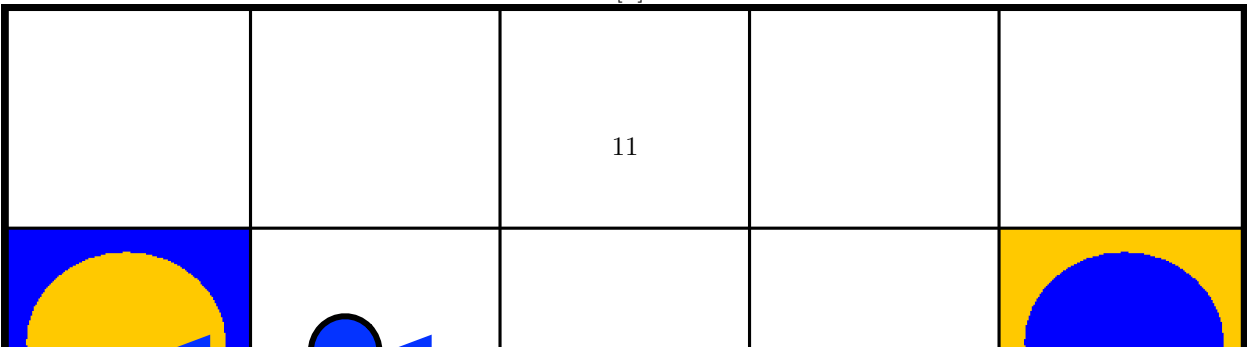
Figure 8

Figure 9

11

Figure 12

12
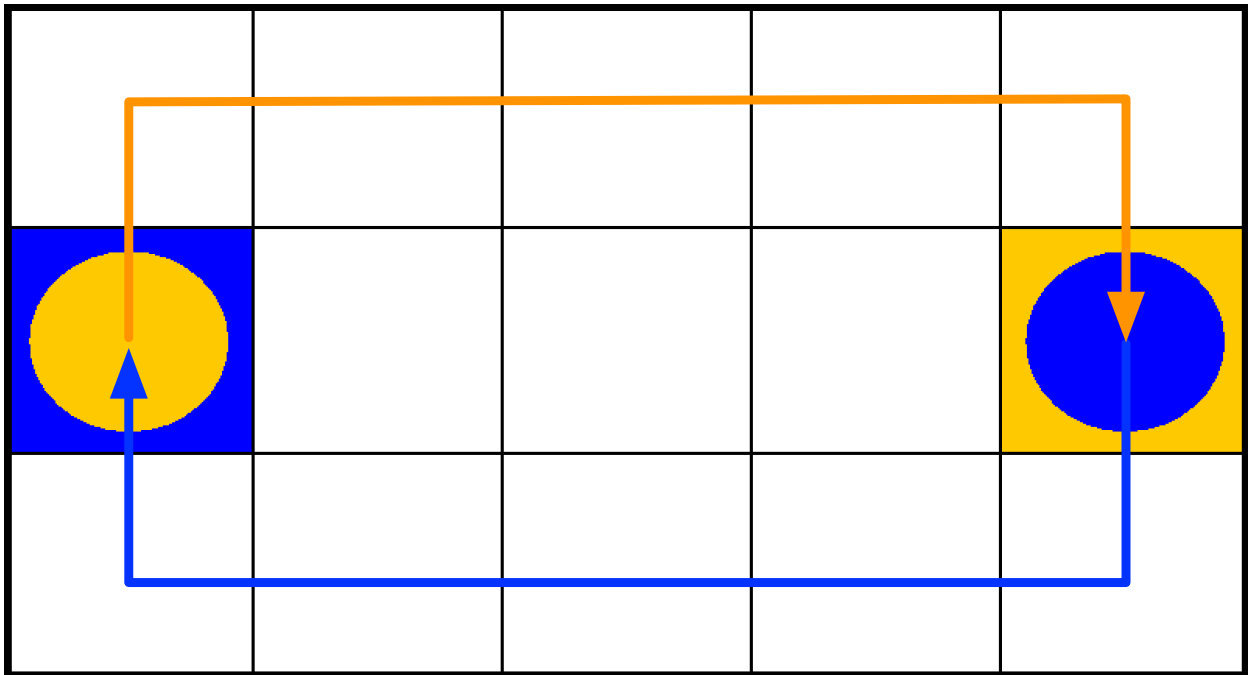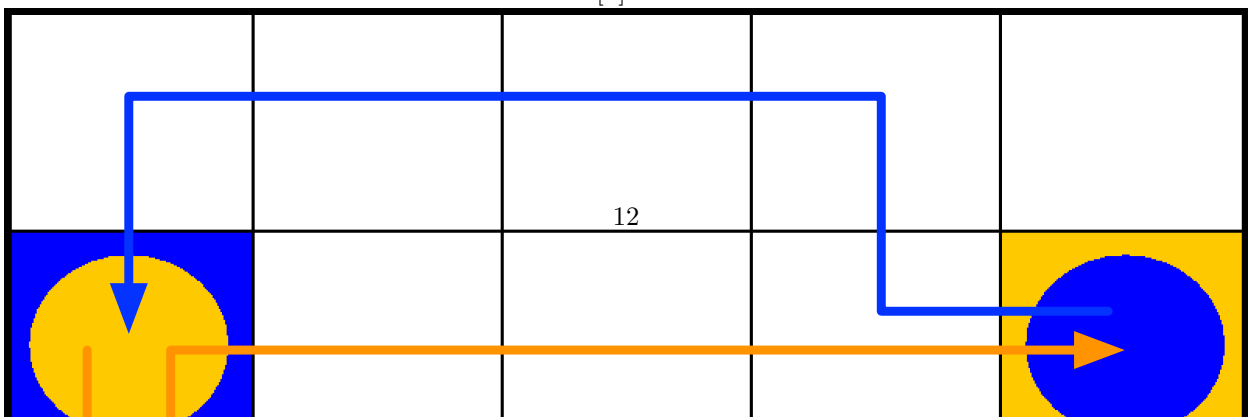
[x=1.7cm,y=1.8cm]
[latent] (R) $\boldsymbol{\theta}$; [obs,
right=of R] (X) A ;
[obs, above=of X] (S)
S ;
SX ; RX ;
traj (S) (X) N ;
data (traj) M ;

Figure 18: Classic
IRL

[x=1.7cm,y=1.8cm]
[latent] (R) $\boldsymbol{\theta}$; [latent,
right=of R] (X) X ;
[obs, above=of X] (S)
S ; [obs, right=of S]
(A) A ;
SX ; AX ; RX ;
traj (S) (A) (X) N ;
data (traj) M ;

Figure 19: SABL La-
bel

[x=1.7cm,y=1.8cm] [la-
tent] (R) $\boldsymbol{\theta}$; [latent,
right=of R] (X) X ;
[obs, below=of X] (L) L
; [obs, above=of X] (S)
S ; [obs, right=of S] (A)
A ;
[above=of   L]L-
fright:Sigmoid ;
SX ; AX ; RX ; XL-fL ;
traj (S) (A) (X) N ;
data (traj)(L-f)(L) M ;
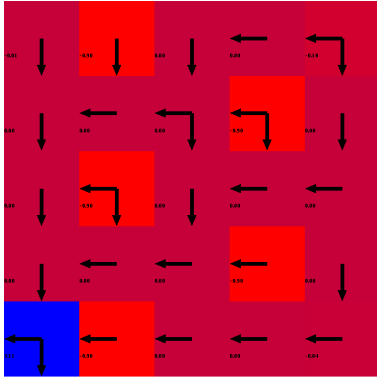
Figure 20: Feedback
IRL

Figure 21: Two positive labelled
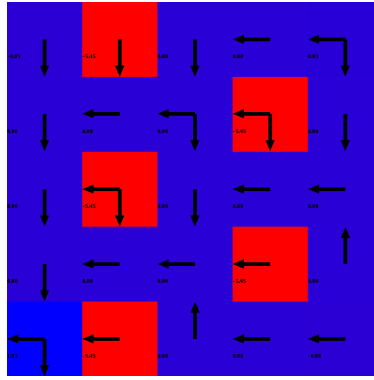trajectories.

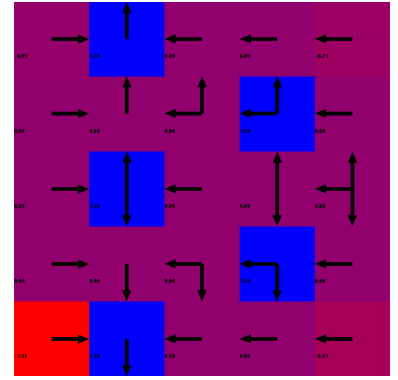Figure 22: Positive and negative
labelled trajectories.

Figure 23: Negative and positive
labelled trajectories.

in different matches, once learned, that equilibrium was consistently employed in all subsequent rounds. Of particular interest are the results in the fifth match (Figure 6e). In the first round of this match, both players wait (a suboptimal joint action), and Blue then moves in an odd looping motion. Nevertheless, the agents learn from the experience, ultimately finding collaborative behavior that works well for them both.

3.3 Feedback IRL

To complement the new computational model we put forth, the new technology that we plan to develop as part of this proposal is a new model of IRL, and accompanying algorithms, which allows for both positive and negative (not necessarily binary) labels, and, moreover, which also allows for partial feedback, meaning subtrajectories (i.e., options [**?**]) can be ascribed such labels.

Nakul: Experiments In this experiment we want to test our labelled inverse reinforcement learning algorithm with simple experiments to see if the agent learns good behavior based on human feedback. The agent is in a world with puddles of multiple colors - red, blue, green, yellow and pink. The agent is allowed north, south, east and west actions. If there is a wall in the way of the agent it does not move at all. For example if the agent tries to move east and it is next to the eastern wall of the grid, it doesn't move at all. For the experiments we will demonstrate two trajectories to the agent, and a pair of labels corresponding to each trajectory. We will show that changes in the labels for the same trajectory lead to changes in the behaviour of the agent.

The first trajectory has the agent move two steps, over the blue puddle to reach the red puddle. The second trajectory has the agent move a single step to the blue puddle. These trajectories are shown in the first set of Figures **??**.

For the first experiment we teach the agent that both behaviours shown with the demonstrated trajectories are good, that is, walking over both puddles is good. Hence, we give a label of +1 and +1 to both trajectories. The policy after learning is shown in Figure **??**. It is clear that the agent prefers to reach the red puddle with the shortest path and stay over it infinitely. The agent doesn't learn to stay over the blue puddle as one of the trajectories was to walk over the blue to reach the red puddle as the goal state. The agent also values the red puddle higher than any cell in the grid.

In the second experiment we teach the agent that walking over blue paddles is bad and getting to the red puddle is good. Hence, we label the first trajecotry +1, and label the second trajectory −1. The output policy in Figure **??** shows that the agent avoids all blue puddle and gets to the red puddle as quickly as possible. Further, the red puddle has the highest value in the grid.

In the third experiment we want to teach the agent that the red puddle should be avoided and the getting to the blue puddle is good. Hence, we label the first trajectory −1 and the second trajectory +1. The output policy in Figure **??** shows that the agent prefers reaching any of the blue puddles in the grid, especially the ones where it can stay in the blue cells indefinitely.

In the fourth experiment we teach the agent that both red and blue puddles are bad. Hence, we label both trajectories −1. The output policy in Figure **??** shows that the agent avoids all the puddles and tries to stay in the white spaces on the grid.

Thus, we can change the agent's behaviour with example trajectories and more importantly the labels. This learning behaviour is important for teaching an agent to play with a human, we the human can critique the agent as they play and lable a round of play good or bad depending on whether the agents reached their co-operative goal. Further, our algorithm allows the teacher to critique individual actions that the agent takes allowing a more minute control of the agent's behaviour.

4   Plan

The three-year timeline for our proposed project is as follows:

- **Year 1**: Complete the development of an experimental testbed that pairs humans with a (single) artificial agent. Demonstrate that the agent can successfully learn collaborative behavior from batches of demonstrations (offline), as well as online while simultaneously interacting with a human.

- **Year 2**: Investigate methods for greater generalization during learning, including methods to generalize collaborations from one state space to another, and learning norms within *populations* of interacting agents instead of just pairs of agents.**Amy: i think populations are beyond the scope of the current grant.**

- **Year 3**: Extend and evaluate our approach in real-world applications such as robotic-human collaborations,**Amy: EXAMPLE!?** and develop any additional methods necessary to handle these scenarios.**Amy: too vague; unconvincing**

Amy: this paragraph is bad. scrap; rewrite. In year one, we will develop evaluation metrics, beginning with experiments evaluating the interactive learning algorithm by comparing how often and how quickly humans are able to arrive at collaborative behavior when playing with another artificial agent vs. another human. For the batch learning component, we will develop new quantitative metrics to assess trajectory similarity.

Amy: this paragraph is good; Turing test. elaborate/play up. We will also develop ways for people to also assess how natural it is to interact with our artificial agents. For example, we could have two humans interact for awhile, and then, without notifying them, swap their partner with an artificial agent trained from the interaction history. Participants could then be asked if they noticed any change, and

behavior would be compared between cases three cases: no swap, swapping in a trained agent, and swapping in an untrained agent.

Amy: i don't think generalization is a year 2 thing; it should be inherent in all IRL, from the start, i believe. the word norm is everywhere. In year two, we will investigate how to transfer and generalize norms**Amy: FIX!** across environments. To facilitate generalization between state spaces, we will start by examining standard features and representations used in existing reinforcement-learning and IRL research (e.g., linear models, neural networks, etc.) and build from them, as necessary, approaches to better capture norms. We will evaluate generalization by first having users learn a norm in one grid. Afterwards, they will transfer to a new grid where a "literal" norm cannot be directly applied (e.g., adding a wall in the path that one agent would normally take), but a more "general" norm can be transfered (e.g., I move above the other agent). We will evaluate these methods using the methods developed in year one.

Amy: all of this should be in the science section; under learning in populations/learning norms. after describing it there, can figure out where it fits in the year-to-year plan. The algorithms described in this proposal are designed for learning among a set of agents all playing in same (one) game. We will extend our work to scenarios where agents play different games, with different subsets of agents drawn from the same population. We expect that a group of people can converge on a norm in a setting like this, and so will extend our algorithms as necessary to support this robust form of norm-learning. To evaluate our extensions, we will compare norm formation in purely human groups to hybrid human-machine groups.

In year three we will test our approach in a real-world domain: robot-human collaboration. Our simulated worlds suggest that our learning agents could collaborate better with people than traditional approaches, but the capabilities of robots may inhibit the effectiveness of our approach (e.g., robots might move slower than people expect). We will evaluate our algorithm in real-world scenarios by comparing its performance at controlling a robot to when another human controls the robot via teleoperation. We expect this to inspire novel problems tuned to real-world problems. For example, computation time may be a limiting factor to the success of our machine agents, and if so, we will develop more efficient implementations.

Experimental Platforms   **Amy: two platforms: BURLAP; MTurk**

The open source Brown-UMBC Reinforcement Learning and Planning (BURLAP) library is the software infrastructure used for algorithm development in this project. Developed by James MacGlashan under the direction of co-PI Littman, the system provides a flexible and powerful implementation of a wide variety of reinforcement-learning-related approaches and has been extensively in both published research and education (including a widely used reinforcement-learning MOOC). The gradient-based inverse reinforcement-learning algorithm MLIRL is the basis for the work we describe in this proposal and it is available as part of the BURLAP distribution.

Following the preliminary work described in this proposal, human-subject experiments will be conducted on the Amazon Mechanical Turk platform.

5   Deliverables

The proposed project includes the following deliverables:

1. An *experimental platform* in which humans can interact with artificial agents, and that thereby provides a space for artificial agents to demonstrate their capacity to represent, learn, and ultimately activate norms in context-appropriate ways.

2. *Software infrastructure for learning together with machine-learning algorithms* that enable the representation, learning, and activation of norms in simulated stochastic games. Code will be made available as a part of the Brown-UMBC Reinforcement Learning and Planning library[8].

---

[8]http://burlap.cs.brown.edu

3. Data comprised of the behaviors produced by both humans and machines in stochastic games, under varying conditions of complexity and uncertainty, as well empirical evaluations of those behaviors. These data will form a set of *computational benchmarks*, which other researchers can freely use to evaluate their own computational models of norms.

4. All of the above for coordination problems, specifically. This will produce a set of *coordination benchmark*, which we hope will inspire other researchers to explore alternative approaches to human-machine collaboration.

In year two, we will organize a workshop on the machine learning of social norms. Such a gathering is timely, as a number of subareas are grappling with related problems. We will include researchers in human-robot collaboration [**?**], intelligent software, end-user programming of household devices [**?**], agent-based bidding [**?**], multi-agent reinforcement learning [**?**], computational social cognition [**?**], and others.

6   Evaluation

Amy: this was not included in the 2015 proposal; content was merged into the 3 year plan section.

We will evaluate our approach in multiple ways. First, we will test our interactive learning algorithm with other human players in our experimental testbed and compare how often and quickly humans are able to arrive at coordinated plan with our artificial agent versus other humans. To test the expressiveness of our algorithms space of learnable solutions, we will also evaluate how many different kinds of strategies can be found when players play with our agent and compare that to the set of strategies that can be found by humans playing with each other. Subjective evaluations from the human participants will be used to assess whether interacting with the agent feels natural.

For the batch learning algorithm, we will develop new quantitative metrics to assess behavior similarity so that trajectories produced from the batch learning can be quantitatively compared to the actual human participants. Additionally, using our experimental test bed, we will have two humans interact for a number of sessions and then, without notification to the users, swap out their partner with an artificial agent trained from the history of interactions they had with their human partner. At the end of all interactions, we will ask the user whether they thought they were playing with the same partner the whole time or if their partner switched. Their subjective response to whether they thought their partner was switched at any point will be compared to their subjective response when their partner is swapped out with an artificial agent that did not perform any learning and when their partner is swapped with a different human player.

Finally, we are also interested in norms generalizing to new state spaces. Using our environmental testbed, we will have participants play in a sequence of different grids and using the same kinds of evaluations listed above, test whether our algorithm can safely generalize to new grids as well as other humans can.

7   Broader Impacts

Recent trends in computer science and artificial intelligence are moving us toward increasing dependence on human–computer collaborative systems in which people and software each make decisions that impact one another. Some fields, such as human–computer interaction (HCI), focus on systems in which a human being is primarily in control and the computer's decisions are assessed in terms of the positive impact they have on the user. An interesting recent example is Centaur chess `http://www.huffingtonpost.com/mike-cassidy/centaur-chess-shows-power_b_6383606.html` in which a human and machine team up to play on the same side in chess, both making decisions, but with the machine acting as an advisor and the human deciding which moves to actually make. Other fields, like human computation and crowd sourcing, combine human and machine expertise with the machine being the ultimate arbiter of behavior and a group of human beings acting as lower level computational components.

True collaboration, however, does not start with one participant being assigned to a leadership role. Instead, the various agents need to dynamically negotiate their roles and jockey for position, discovering when and how to trust each other to move forward. A concrete example is in the context of self-driving cars. Cars share the road with each other and must carefully choose when to be responsive to other vehicles and when to assert themselves to create a situation that benefits them. Doing so makes a significant difference in the driving's effectiveness [**?**]. A related problem arises when a self-driving wheelchair attempts to move through a group of pedestrians [**?**]. More generally, robots that interact with people in the physical world need to navigate the complex give-and-take of establishing mutually beneficial behaviors where possible.

"Social autonomous driving" class for undergraduates building on the Duckietown platform[9].

Amy: 2015: The domains that would be positively impacted by machines that can collaborate effectively with humans is endless, such as robotic sous chefs and gerontechnological support for aging in place. In particular, the results of this project synergize with the main efforts of Brown University's Humanity Centered Robotics Initiative (HCRI), of which co-PI Littman is co-director. Specifically, within HCRI there are ongoing efforts to design robotic systems that support independent living tasks. For an elderly person to trust and collaborate on tasks with a machine effectively, the machine must act in a manner that the elderly person expects. This project lies the foundation for these important applications.

In an effort to increase diversity in computer science, we have pre-selected two graduate students to participate in this project—one woman, and the other Hispanic. All students (both graduate and under-graduate) who join our team will benefit from collaborating with the cognitive psychologists on our team. Specifically, they will strengthen their understanding of a number of fields, all of which are critical to the development of artificial agents that collaborate effectively with humans: e.g., behavioral economics, cognitive psychology, reinforcement learning, and software engineering.

"One of the things that [President Obama] really strongly believe[s] in is that we need to have more girls interested in math, science, and engineering." To that end, we also plan to integrate our work on this project into Artemis, a free summer program run at Brown and directed by PI Greenwald that introduces rising 9th grade girls to computational thinking. For example, we might have the Artemis girls teach a robot to collaborate with them on various tasks, such as finding and navigating to common household objects.

Our deliverables include an open-source publicly accessible toolkit for agents to learn collaborative behavior via reinforcement learning. Further, we will build a database of machine-machine, human-machine, and human-human experimental results on collaborative-learning problems, which can serve as a benchmark for future researchers to build artifical agents that increasingly achieve human-like behavior. Finally, we expect to publish the results of the proposed research in top-tier archival, conference proceedings and journals with high impact factors, and present our work at both computer science and psychology conferences.

8   Results from Prior NSF Support

Intellectual Merit   Greenwald is currently funded by NSF (RI: Small–1217761, 2012–2016, $450k) to build artificial agents that assist humans with decision making in information-rich and time-critical environments like online markets. This work is ongoing, but some preliminary publications include [**?, ?, ?, ?**]. Previously, she was funded to develop "Methods of Empirical Mechanism Design (EMD)" (CCF: Medium–0905234, 2009–2012, $850k, with Mike Wellman). This project expanded the scope of mechanism design beyond the small-scale, stylized, or idealized domains most previous work was limited to. Publications include [**?, ?, ?**]. Before that, she received two prior NSF awards, a PECASE CAREER grant, "Computational Social Choice Theory" (IIS: Career–0133689, 2002–2007, $375k), and "Efficient Link Analysis" (IIS: Small–0534586, 2005–2008, $363k), which focused on ranking web pages and other social networks with hierarchical structure.

Littman is a co-PI on "RI: Medium: Collaborative Research: Teaching Computers to Follow Verbal Instructions" (No. 1065195, 9/11–8/14, $704K) and "RI: Small: Collaborative Research: Speeding Up Learning through Modeling the Pragmatics of Training" (No. 1319305, 10/13–9/15, $440k). These proposals developed autonomous agents that extract preferences from people via verbal interaction and reward feedback [**?, ?, ?**].

---

[9]http://duckietown.mit.edu/

Broader Impact   Broader impacts of our past work include undergraduate and graduate student training, design and evaluation of learning modules for outreach programs Learning Exchange and Artemis, and organizing two major computer science conferences (Littman) [**?**]. We also published novel benchmarks for evaluating grounded language learning programs (Littman), and developed a simulation platform for empirical game-theoretic analysis (Greenwald).

Greenwald also had an additional NSF award (EAGER: 1059570, 2010–2012, $90k) which supported the expansion and evaluation of the Artemis Project, a free, summer program in which Brown computer science women teach computer science to rising 9th grade girls. This program has the dual effect of empowering Brown women, while at the same time, exposing girls to computer science. With this money, she successfully expanded Artemis to BU, and ran pilot programs at Columbia and UMBC.

Greenwald also has $5k in funding from the Tides Foundation, through the IgniteCS progam, for a project entitled Bringing Computer Science Education to Providence Public Schools.