# Maximum Causal Entropy Specification Inference from Demonstrations

Marcell Vazquez-Chanlatte and Sanjit A. Seshia

University of California, Berkeley, USA

**Abstract.** In many settings, such as robotics, demonstrations provide a natural way to specify tasks. However, most methods for learning from demonstrations either do not provide guarantees that the learned artifacts can be safely composed or do not explicitly capture temporal properties. Motivated by this deficit, recent works have proposed learning Boolean *task specifications*, a class of Boolean non-Markovian rewards which admit well-defined composition and explicitly handle historical dependencies. This work continues this line of research by adapting maximum *causal* entropy inverse reinforcement learning to estimate the posteriori probability of a specification given a multi-set of demonstrations. The key algorithmic insight is to leverage the extensive literature and tooling on reduced ordered binary decision diagrams to efficiently encode a time unrolled Markov Decision Process. This enables transforming a naïve algorithm with running time exponential in the episode length, into a polynomial time algorithm.

## 1 Introduction

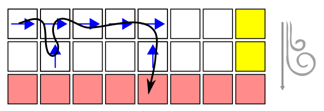


Fig. 1: Example of an agent unsuccessfully demonstrating the task "reach a yellow tile while avoiding red tiles" .

In many settings, episodic demonstrations provide a natural and robust mechanism to partially specify a task, even in the presence of errors. For example, consider the agent operating in the gridworld illustrated in Fig. 1. Blue arrows denote intended actions and the solid black arrow shows the agent's actual path. This path can stochastically differ from the blue arrows due to a downward wind. One might naturally ask: "What task was this agent attempting to perform?" Even without knowing if this was a positive or negative example, based on the agent's state/action sequence, one can reasonably infer the agent's intent, namely, "reach the yellow tile while avoiding the red tiles." Compared with traditional learning from positive and negative examples, this is somewhat surprising, particularly given that the task is never actually demonstrated in Fig. 1.

This problem, inferring intent from demonstrations, has received a fair amount of attention over the past two decades particularly within the robotics community [22][33][5][30]. In this literature, one traditionally models the demonstrator as operating within a dynamical system whose transition relation only depends on the current state and action (called the Markov condition). However, even if the dynamics are Markovian, many tasks are naturally modeled in history dependent (non-Markovian) terms, e.g., "if the robot enters a blue tile, then it must touch a brown tile *before* touching a yellow tile". Unfortunately, most methods for learning from demonstrations either do not provide guarantees that the learned artifacts (e.g. rewards) can be safely composed or do not explicitly capture history dependencies [30].

Motivated by this deficit, recent works have proposed specializing to **task specifications**, a class of Boolean non-Markovian rewards induced by formal languages. This additional structure admits well-defined compositions and explicitly captures temporal dependencies [15][30]. A particularly promising direction has been to adapt maximum entropy inverse reinforcement learning [33] to task specifications, enabling a form of robust specification inference, even in the presence unlabeled demonstration errors [30].

However, while powerful, the principle of maximum entropy is limited to settings where the dynamics are deterministic or agents that use open-loop policies [33]. This is because the principle of maximum entropy incorrectly allows the agent's predicted policy to depend on future state values resulting in an overly optimistic agent [19]. For instance, in our gridworld example (Fig. 1), the principle of maximum entropy would discount the possibility of slipping, and thus we would not forecast the agent to correct its trajectory after slipping once.

This work continues this line of research by instead using the principle of maximum *causal* entropy, which generalizes the principle of maximum entropy to general stochastic decision processes [32]. While a conceptually straightforward extension, a naïve application of maximum *causal* entropy inverse reinforcement learning to non-Markovian rewards results in an algorithm with run-time exponential in the episode length, a phenomenon sometimes known as the **curse of history** [24]. The key algorithmic insight in this paper is to leverage the extensive literature and tooling on Reduced Ordered Binary Decision Diagrams (BDDs) [3] to efficiently encode the time unrolled composition of the dynamics and task specification. This allows us to translate a naïve exponential time algorithm into a polynomial time algorithm. In particular, we shall show that this BDD has size at most linear in the episode length making inference comparatively efficient.

**1.1 Related Work:** Our work is intimately related to the fields of Inverse Reinforcement Learning and Grammatical Inference. **Grammatical inference** [8] refers to the well-developed literature on learning a formal grammar (often an automaton) from data. Examples include learning the smallest automata that in consistent with a set of positive and negative strings [7][8] or learning an automaton using membership and equivalence queries [1]. This and related work can be seen as extending these methods to unlabeled and potentially noisy demonstrations, where demonstrations differ from examples due to the existence of a dynamics model. This notion of demonstration derives from the Inverse Reinforcement Learning literature.

In **Inverse Reinforcement Learning** (IRL) [22] the demonstrator, operating in a stochastic environment, is assumed to attempt to (approximately) optimize some unknown reward function over the trajectories. In particular, one traditionally assumes a trajectory's reward is the sum of state rewards of the trajectory. This formalism offers a succinct mechanism to encode and generalize the goals of the demonstrator to new and unseen environments.

In the IRL framework, the problem of learning from demonstrations can then be cast as a Bayesian inference problem [25] to predict the most probable reward function. To make this inference procedure well-defined and robust to demonstration/modeling noise, Maximum Entropy [33] and Maximum Causal Entropy [32] IRL appeal to the principles of maximum entropy [13] and maximum causal entropy respectively [32]. This results in a likelihood over the demonstrations which is no more committed to any particular behavior than what is required to match observed statistical features, e.g., average distance to an obstacle. While this approach was initially limited to rewards represented as linear combinations of scalar features, IRL has been successfully adapted to arbitrary function approximators such as Gaussian processes [20] and neural networks [5]. As stated in the introduction, while powerful, traditional IRL provides no principled mechanism for composing the resulting rewards.

**Compositional RL:** To address this deficit, composition using soft optimality has recently received a fair amount of attention; however, the compositions are limited to either strict disjunction (do X *or* Y) [26] [27] or conjunction (do X *and* Y) [6]. Further, this soft optimality only bounds the deviation from simultaneously optimizing both rewards. Thus, optimizing the composition does not preclude violating safety constraints embedded in the rewards (e.g., do not enter the red tiles).

**Logic based IRL:** Another promising approach for introducing composationality has been the recent research on automata and logic based encodings of rewards [11][14] which admit well defined compositions. To this end, work has been done on inferring Linear Temporal Logic (LTL) formulas by finding the specification that minimizes the expected number of violations by an optimal agent compared to the expected number of violations by an agent applying actions uniformly at random [15]. The computation of the optimal agent's expected violations is done via dynamic programming on the explicit product of the deterministic Rabin automaton [4] of the specification and the state dynamics. A fundamental drawback of this procedure is that due to the curse of history, it incurs a heavy run-time cost, even on simple two state and two action Markov Decision Processes. Additionally, as with early work on grammatical inference and IRL, these techniques do not produce likelihood estimates amenable to Bayesian inference.

**Maximum Entropy Specification Inference:** In our previous work [30], we adapted maximum entropy IRL to learn task specifications. Similar to standard maximum entropy IRL, this technique produces robust likelihood estimates. However, due to the use of the principle of maximum entropy, rather than maximum *causal* entropy, this model is limited to settings where the dynamics are deterministic or agents with open-loop policies [33].

**Inference using BDDs:** This work makes heavy use of Binary Decision Diagrams (BDDs) [3] which are frequently used in symbolic value iteration for Markov Decision Processes [9] and reachability analysis for probabilistic systems[18]. However, the literature has largely relied on Multi-Terminal BDDs to encode the transition probabilities for a **single** time step. In contrast, this work introduces a two-terminal encoding based on the finite unrolling of a probabilistic circuit. To the best of our knowledge, the most similar usage of BDDs for inference appears in the independently discovered literal weight based encoding of [10] - although their encoding does not directly support non-determinism or state-indexed random variables.

**Contributions:** The primary contributions of this work are two fold. First, we leverage the principle of maximum causal entropy to provide the likelihood of a specification given a set of demonstrations. This formulation removes the deterministic and/or open-loop restriction imposed by prior work based on the principle of maximum entropy. Second, to mitigate the curse of history, we propose using a BDD to encode the time unrolled Markov Decision Process that the maximum causal entropy forecaster is defined over. We prove that this BDD has size that grows linearly with the horizon and quasi-linearly with the number of actions. Furthermore, we prove that our derived likelihood estimates are robust to the particular reward associated with satisfying the specification. Finally, we provide an initial experimental validation of our method. An overview of this pipeline is provided in Fig. 8.

## 2 Problem Setup

We seek to learn task specifications from demonstrations provided by a teacher who executes a sequence of actions that probabilistically change the system state. For simplicity, we assume that the set of actions and states are finite and fully observed. Further, until Sec. 5.3, we shall assume that all demonstrations are a fixed length, $\tau \in \mathbb{N}$. Formally, we begin by modeling the underlying dynamics as a probabilistic automaton.

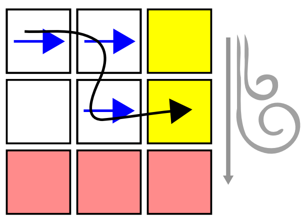
**Definition 1** *A* ***probabilistic automaton*** *(PA) is a tuple* $(S, s_0, A, \delta)$*, where* $S$ *is the finite set of states,* $s_0 \in S$ *is the initial state,* $A$ *is a finite set of actions, and* $\delta$ *specifies the transition probability of going from state* $s$ *to state* $s'$ *given action* $a$*, i.e.* $\delta(s, a, s') = \Pr(s' \mid s, a)$*.*

*A* ***trace***[a]*,* $\xi$*, is a sequence of (action, state) pairs implicitly starting from* $s_0$*. A trace of length* $\tau \in \mathbb{N}$ *is an element of* $(A \times S)^\tau$*.*

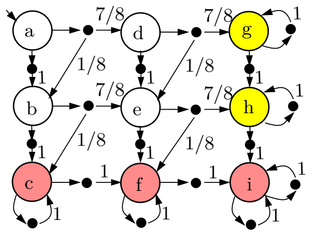
[a] sometimes referred to as a trajectory or behavior.

Note that probabilistic automata are equivalently characterized as *1½ player games* where each round has the agent choose an action and then the environment samples a state transition outcome. In fact, this alternative characterization is implicitly encoded in the directed bipartite graph used to visualize probabilistic automata (see Fig 2b). In this language, we refer to the nodes where the agent makes a decision as a **decision node** and the nodes where the environment samples an outcome as a **chance node**.

Next, we develop machinery to distinguish between desirable and undesirable traces. For simplicity, we focus on finite trace properties, referred to as specifi-



(a) Example trajectory in a gridworld where the agent can attempt to move right and down, although with a small probability the wind will move the agent down, independent of the action.



(b) PA describing the dynamics of Fig 2a as a 1½ player game. The large circles indicate states (agent decisions) and the small black circles denote the environment response probabilities.

Fig. 2: Example of gridworld probabilistic automata (PA).

cations, that are decidable within some fixed $\tau \in \mathbb{N}$ time steps, e.g., "Recharge before t=20."

**Definition 2** *A* ***task specification****, $\varphi$, (or simply* ***specification****) is a subset of traces. For simplicity, we shall assume that each trace is of a fixed length $\tau \in \mathbb{N}$, e.g.,*

$$\varphi \subseteq (A \times S)^\tau \tag{1}$$

*A collection of specifications, $\Phi$, is called a* ***concept class****. Further, we define true $\overset{\text{def}}{=} (A \times S)^\tau$, $\neg\varphi \overset{\text{def}}{=}$ true $\setminus \varphi$, and false $\overset{\text{def}}{=} \neg$true.*

Often specifications are not directly given as sets, but induced by abstract descriptions of a task. For example, the task "avoid lava" induces a concrete set of traces that never enter lava tiles. If the workspace/world/dynamics change, this abstract specification would map to a different set of traces.

**2.1 Specification Inference from Demonstrations:** The primary task in this paper is to find the specification that best explains/forecasts the behavior of an agent. As in our prior work [30], we formalize our problem statement as:

**Definition 3** *The* ***specification inference from demonstrations*** *problem is a tuple $(M, X, \Phi, D)$ where $M = (S, s_0, A, \delta)$ is a probabilistic automaton, $X$ is a (multi-)set of $\tau$-length traces drawn from an unknown distribution induced by a teacher attempting to demonstrate (satisfy) some unknown task specification within $M$, $\Phi$ is a concept class of specifications, and $D$ is a prior distribution over $\Phi$. A solution to $(M, X, \Phi, D)$ is:*

$$\varphi^* \in \underset{\varphi \in \Phi}{\arg\max} \Pr(X \mid M, \varphi) \cdot \Pr_{\varphi \sim D}(\varphi) \tag{2}$$

*where $\Pr(X \mid M, \varphi)$ denotes the likelihood that the teacher would have demonstrated $X$ given the task $\varphi$.*

Of course, by itself, the above formulation is ill-posed as $\Pr(X \mid M, \varphi)$ is left undefined. Below, we shall propose leveraging Maximum Causal Entropy Inverse Reinforcement Learning (IRL) to select the demonstration likelihood distribution in a regret minimizing manner.

## 3 Leveraging Inverse Reinforcement Learning

The key idea of Inverse Reinforcement Learning (IRL), or perhaps more accurately Inverse Optimal Control, is to find the reward structure that best explains the actions of a reward optimizing agent operating in a Markov Decision Process. We formalize below.

**Definition 4** *A* ***Markov Decision Process (MDP)*** *is a probabilistic automaton endowed with a* ***reward map*** *from states to reals,* $r : S \to \mathbb{R}$*. This reward mapping is lifted to traces via,*

$$R(\xi) \overset{\text{def}}{=} \sum_{s \in \xi} r(s). \tag{3}$$

*Remark 1.* Note that a temporal discount factor, $\gamma \in [0, 1]$ can be added into (3) by introducing a sink state, \$, to the MDP, where $r(\$) = 0$ and

$$\Pr(s' = \$ \mid s, a) = \begin{cases} \gamma & \text{if } s \neq \$ \\ 1 & \text{otherwise} \end{cases}. \tag{4}$$

Given a MDP, the goal of an agent is to maximize the expected trace reward. In this work, we shall restrict ourselves to rewards that are given as a linear combination of **state features**, $\mathbf{f} : S \to \mathbb{R}^n_{\geq 0}$, e.g.,

$$r(s) = \theta \cdot \mathbf{f}(s) \tag{5}$$

for some $\theta \in \mathbb{R}^n$. Note that since state features can themselves be rewards, such a restriction does not actually restrict the space of possible rewards.

*Example 1.* Let the components of $\mathbf{f}(s)$ be distances to various locations on a map. Then the choice of $\theta$ characterizes the relative preferences in avoiding/reaching the respective locations.

Formally, we model an agent as acting according to a **policy**.

**Definition 5** *A policy,* $\pi$*, is a state indexed distribution over actions,*

$$\Pr(a \mid s) = \pi(a \mid s). \tag{6}$$

In this language, the agent's goal is equivalent to finding a policy which maximizes the expected trace reward. We shall refer to a trace generated by such an agent as a **demonstration**. Due to the Markov requirement, the likelihood of a demonstration, $\xi$, given a particular policy, $\pi$, and probabilistic automaton, $M$, is easily stated as:

$$\Pr(\xi \mid M, \pi) = \prod_{s', a, s \in \xi} \Pr(s' \mid s, a) \cdot \Pr(a \mid s). \tag{7}$$

Thus, the likelihood of multi-set of i.i.d demonstrations, $X$, is given by:

$$\Pr(X \mid M, \pi) = \prod_{\xi \in X} \Pr(\xi \mid M, \pi). \tag{8}$$

**3.1 Inverse Reinforcement Learning (IRL):** As previously stated, the main motivation in introducing the MDP formalism has been to discuss the inverse problem. Namely, given a set of demonstrations, find the reward that best "explains" the agent's behavior, where by "explain" one typically means that under the conjectured reward, the agent's behavior was approximately optimal. Notice however, that many undesirable rewards satisfy this property. For example, consider the following reward in which every demonstration is optimal,

$$r : s \mapsto 0. \tag{9}$$

Furthermore, observe that given a fixed reward, many policies are approximately optimal! For instance, using (9), an optimal agent could pick actions uniformly at random or select a single action to always apply.

**3.2 Maximum Causal Entropy IRL:** A popular, and in practice effective, solution to the lack of unique policy conundrum is to appeal to the **principle of maximum causal entropy** [32]. To formalize this principle, we recall the definitions of causally conditioned probability [17] and causal entropy [17][23].

**Definition 6** *Let $X_{1:\tau} \overset{\text{def}}{=} X_1, \ldots, X_\tau$ denote a temporal sequence of $\tau \in \mathbb{N}$ random variables. The probability of a sequence $Y_{1:\tau}$* ***causally conditioned*** *on sequence $X_{1:\tau}$ is:*

$$\Pr(Y_{1:\tau} \mid\mid X_{1:\tau}) \overset{\text{def}}{=} \prod_{t=1}^{\tau} \Pr(Y_t \mid X_{1:t}, Y_{1:t-1}) \tag{10}$$

*The* ***causal entropy*** *of $Y_{1:\tau}$ given $X_{1:\tau}$ is defined as,*

$$H(Y_{1:\tau} \mid\mid X_{1:\tau}) \overset{\text{def}}{=} \mathop{\mathbb{E}}_{Y_{1:\tau}, X_{1:\tau}} \left[-\log(\Pr(Y_{1:\tau} \mid\mid X_{1:\tau}))\right] \tag{11}$$

In the case of inverse reinforcement learning, the principle of maximum causal entropy suggests forecasting using the policy whose action sequence, $A_{1:\tau}$, has the highest causal entropy, conditioned on the state sequence, $S_{1:\tau}$. That is, find the policy that maximizes

$$H(A_{1:\tau} \mid\mid S_{1:\tau}), \tag{12}$$

subject to feature matching constraints, $\mathbb{E}[\mathbf{f}]$, e.g., does the resulting policy, $\pi^*$, complete the task as seen in the data. Compared to all other policies, this policy

(i) minimizes regret with respect to model/reward uncertainty, (ii) ensures that the agent's predicted policy does not depend on the future, (iii) is consistent with observed feature statistics [32].

Concretely, as proved in [32], when an agent is attempting to maximize the sum of feature state rewards, $\sum_{t=1}^{T} \theta \cdot \mathbf{f}(s_t)$, the principle of maximum causal entropy prescribes the following policy:

**Maximum Causal Entropy Policy:**

$$\log\big(\pi_\theta(a_t \mid s_t)\big) \overset{\text{def}}{=} Q_\theta(a_t, s_t) - V_\theta(s_t) \tag{13}$$

where

$$\begin{aligned} Q_\theta(a_t, s_t) &\overset{\text{def}}{=} \mathop{\mathbb{E}}_{s_{t+1}}\left[V_\theta(s_{t+1}) \mid s_t, a_t\right] + \theta \cdot \mathbf{f}(s_t) \\ V_\theta(s_t) &\overset{\text{def}}{=} \ln \sum_{a_t} e^{Q_\theta(a_t, s_t)} \overset{\text{def}}{=} \text{softmax}_{a_t} Q_\theta(a_t, s_t). \end{aligned} \tag{14}$$

where, $\theta$ is such that (14) results in a policy which matches feature demonstrations.

*Remark 2.* Note that replacing softmax with max in (14) yields the standard Bellman Backup [2] used to compute the optimal policy in tabular reinforcement learning. Further, it can be shown that maximizing causal entropy corresponds to believing that the agent is exponentially biased towards high reward policies [32]:

$$\Pr(\pi_\theta \mid M) \propto \exp\Big(\mathop{\mathbb{E}}_{\xi}[R_\theta(\xi) \mid \pi_\theta, M]\Big), \tag{15}$$

where (14) is the most likely policy under (15).

*Remark 3.* In the special case of scalar state features, $\mathbf{f} : S \to \mathbb{R}_{\geq 0}$, the maximum causal entropy policy (14) becomes increasingly optimal as $\theta \in \mathbb{R}$ increases (since softmax monotonically approaches max). In this setting, we shall refer to $\theta$ as the agent's **rationality coefficient**.

**3.3 Non-Markovian Rewards:** The MDP formalism traditionally requires that the reward map be Markovian (i.e., state based); however, in practice, many tasks are history dependent, e.g. touch a red tile and then a blue tile.

A common trick within the reinforcement learning literature is to simply change the MDP and add the necessary history to the state so that the reward is Markovian, e.g. a flag for touching a red tile. However, in the case of inverse reinforcement learning, by definition, one does not know what the reward is. Therefore, one cannot assume to a priori know what history suffices.

Further exacerbating the situation is the fact that naïvely including the entire history into the state results in an exponential increase in the number of states. Nevertheless, as we shall soon see, by restricting the class of rewards to represent task specifications, this curse can be mitigated to only result in a blow-up that is at most **linear** in the state space size and in the trace length!

To this end, we shall find it fruitful to develop machinery for embedding the full trace history into the state space. Explicitly, we shall refer to the process of adding all history to a probabilistic automaton's (or MDP's) state as **unrolling**.

**Definition 7** *Let $M = (S, s_0, A, \delta)$ be a PA. The* ***unrolling*** *of $M$ is a PA, $M' = (S', s_0, A, \delta')$, where*

$$S' = \{s_0\} \times \bigcup_{i=0}^{\infty} (A \times S)^i \qquad \delta'(\xi_{n+1}, a, \xi_n) = \delta(s_{n+1}, a, s_n)$$
$$\xi_n = \Big(s_0, \ldots, (a_{n-1}, s_n)\Big) \qquad \xi_{n+1} = \Big(s_0, \ldots, (a_n, s_{n+1})\Big) \tag{16}$$

If $R : S^\tau \to \mathbb{R}$ is a non-Markovian reward over $\tau$-length traces, then we endow the corresponding unrolled PA with the now Markovian Reward,

$$r'\Big(s_0, \ldots, (a_{n-1}, s_n)\Big) \overset{\text{def}}{=} \begin{cases} R(s_0, \ldots, s_n) & \text{if } n = \tau \\ 0 & \text{otherwise} \end{cases}. \tag{17}$$

Further, by construction the reward is Markovian in $S'$ and only depends only $\tau$-length state sequences,

$$\sum_{t=0}^{\infty} r'\big((s_0, a_0), \ldots s_\tau\big) = R(s_0, \ldots, s_\tau). \tag{18}$$

Next, observe that for $\tau$-length traces, the 1½ player game formulation's bipartite graph forms a tree of depth $\tau$ (see Fig 3). Further, observe that each leaf corresponds to unique $\tau$-length trace. Thus, to each leaf, we associate the corresponding trace's reward, $R(\xi)$. We shall refer to this tree as a **decision tree**, denoted $\mathbb{T}$.

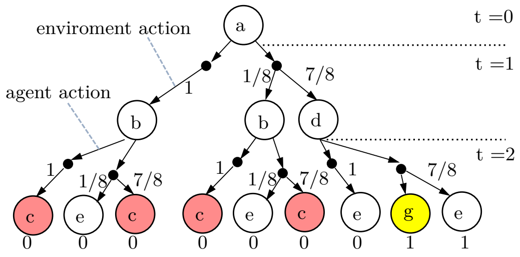


Fig. 3: Decision tree generated by the PA shown in Fig. 2 and specification "By $\tau = 2$, reach a yellow tile while avoiding red tiles.". Here a binary reward is given depending on whether or not the agent satisfies the specification.

Finally, observe that the trace reward depends only on the sequence of agent actions, $A$, and environment actions, $A_e$. That is, $\mathbb{T}$ can be interpreted as a function:

$$\mathbb{T} : (A \times A_e)^\tau \to \mathbb{R}. \tag{19}$$

**3.4 Specifications as Non-Markovian Rewards:** Next, with the intent to frame our specification inference problem as an inverse reinforcement learning problem, we shall overload notation and denote by $\varphi$ the following non-Markovian reward corresponding to a specification $\varphi \in (A \times S)^\tau$,

$$\varphi(\xi) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } \xi \in \varphi \\ 0 & \text{otherwise} \end{cases}. \tag{20}$$

Note that the corresponding decision tree is then a Boolean predicate:

$$\mathbb{T}_\varphi : (A \times A_e)^\tau \to \{0,1\}. \tag{21}$$

**3.5 Computing Maximum Causal Entropy Specification Policies:** Now let us return to the problem of computing the policy prescribed by (14). In particular, note that viewing the unrolled reward (17) as a scalar state feature results in the following soft-Bellman Backup:

$$\begin{aligned} Q_\theta(a_t, \xi_t) &= \mathbb{E}\left[V_\theta(s_{t+1}) \mid \xi_t, a_t\right] \\ V_\theta(\xi_t) &= \begin{cases} \theta \cdot \varphi(\xi_t) & \text{if } t = \tau \\ \text{softmax}_{a_t} Q_\theta(a_t, \xi_t) & \text{otherwise} \end{cases}, \end{aligned} \tag{22}$$

where $\xi_i \in \{s_0\} \times (A \times S)^i$ denotes a state in the unrolled MDP.

Eq (22) thus suggests a naïve dynamic programming scheme over $\mathbb{T}$ starting at the $t = \tau$ leaves to compute $Q_\theta$ and $V_\theta$ (and thus $\pi_\theta$).

Namely, in $\mathbb{T}$, the chance nodes, which correspond to action/state pairs, are responsible for computing $Q$ values and the decision nodes, which correspond to states waiting for an action to be applied, are responsible for computing $V$ values. For chance nodes this is done by taking the softmax of the values of the child nodes. Similarly, for decision nodes, this is done by taking a weighted average of the child nodes, where the weights correspond to the probability of a given transition. This, at least conceptually, corresponds to transforming $\mathbb{T}$ into a bipartite computation graph (see Fig 4).

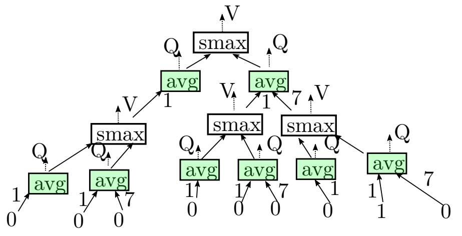


Fig. 4: Computation graph generated from applying (14) to the decision tree shown in Fig 3. Here smax and avg denote the softmax and weighted average respectively.

Next, note that (i) the above dynamic programming scheme can be trivially modified to compute the expected trace reward of the maximum causal entropy policy and (ii) the expected reward increases [1] with the rationality coefficient $\theta$.

Observe then that, due to monotonicity, bisection (binary search) approximates $\theta$ to tolerance $\epsilon$ in $O(\log(1/\epsilon))$ time. Additionally, notice that the likelihood of each demonstration can be computed by traversing the path of length $\tau$ in $\mathbb{T}$ corresponding to the trace and multiplying the corresponding policy and transition probabilities (8). Therefore, if $|A_e| \in \mathbb{N}$ denotes the maximum number of

[1] Formally, this is due to (a) softmax and average being monotonic (b) trajectory rewards only increasing with $\theta$, and (c) $\pi$ exponentially biasing towards high Q-values.

outcomes the environment can choose from (i.e, the branching factor for chance nodes), it follows that the run-time of this naïve scheme is:

$$O\Bigg(\overbrace{\underbrace{\Big(|A|\cdot|A_e|\Big)^{\tau}}_{|\mathbb{T}|}\cdot\underbrace{\log(1/\epsilon)}_{\text{Feature Matching}}}^{\text{compute policy}}+\underbrace{\tau|X|}_{\text{evaluate demos}}\Bigg). \tag{23}$$

**3.6 Task Specification Rewards:** Of course, the problem with this naïve approach is that explicitly encoding the unrolled tree, $\mathbb{T}$, results in an exponential blow-up in the space and time complexity. The key insight in this paper is that the additional structure of task specifications enables avoiding such costs while still being expressive. In particular, as is exemplified in Fig 4, the computation graphs for task specifications are often highly redundant and apt for compression.

In particular, we shall apply the following two semantic preserving transformations: (i) Eliminate nodes whose children are isomorphic sub-graphs, i.e., inconsequential decisions (ii) Combine all isomorphic sub-graphs i.e., equivalent decisions. We refer to the limit of applying these two operations as a **reduced ordered probabilistic decision diagram** and shall denote[2] the reduced variant of $\mathbb{T}$ as $\mathcal{T}$.

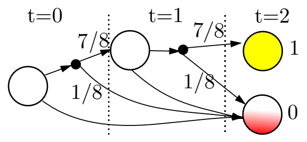


Fig. 5: Reduction of the decision tree shown in Fig. 3.

*Remark 4.* For those familiar, we emphasize that these decision diagrams are MDPs, not Binary Decision Diagrams (see Sec. 4). Importantly, more than two actions can be taken from a node if $\max(|A|, |A_e|) \geq 2$ and $A_e$ has a state dependent probability distribution attached to it. That said, the above transformations are **exactly** the reduction rules for BDDs [3].

As Fig 5 illustrates, reduced decision diagrams can be much smaller than their corresponding decision tree. Nevertheless, we shall briefly postpone characterizing $|\mathcal{T}|$ until developing some additional machinery in Sec. 4. Computationally, three problems remain.

1. How can our naïve dynamic programming scheme be adapted to this compressed structure. In particular, because many interior nodes have been eliminated, one must take care when applying (22).
2. How do concrete demonstrations map to paths in the compressed structure when evaluating likelihoods (8).
3. How can one construct $\mathcal{T}$ without first constructing $\mathbb{T}$, since failing to do so would negate any complexity savings.

We shall postpone discussing solutions to the second and third problems until Sec. 4. The first problem however, can readily be addressed with the tools at hand. Recall that in the variable ordering, nodes alternate between decision and chance nodes (i.e., agent and environment decisions), and thus alternate

[2] Mnemonic: $\mathcal{T}$ is a (typographically) slimmed down variant of $\mathbb{T}$

between taking a softmax and expectations of child values in (22). Next, by definition, if a node is skipped in $\mathcal{T}$, then it must have been inconsequential. Thus the trace reward must have been independent of the decision made at that node. Therefore, the softmax/expectation's corresponding to eliminated nodes must have been over a constant value - otherwise the eliminated sequences would be distinguishable w.r.t $\varphi$. The result is summarized in the following identities, where $\alpha$ denotes the value of an eliminated node's children.

$$\text{softmax}(\overbrace{\alpha, \ldots, \alpha}^{|A|}) = \log(e^{\alpha} + \ldots + e^{\alpha}) = \ln(|A|) + \alpha \tag{24}$$

$$\mathop{\mathbb{E}}_{x}[\alpha] = \sum_{x} p(x)\alpha = \alpha \tag{25}$$

Of course, it could also be the case that a sequence of nodes is skipped in $\mathcal{T}$. Using (24), one can compute the change in value, $\Delta$, that the eliminated sequence of $n$ decision nodes and any number of chance nodes would have applied in $\mathbb{T}$:

$$\Delta(n, \alpha) = \ln(|A|^n) + \alpha = n\ln(|A|) + \alpha \tag{26}$$

Crucially, evaluation of this compressed computation graph is linear in $|\mathcal{T}|$ which as shall later prove, is often much smaller than $|\mathbb{T}|$.

## 4 Constructing and Characterizing $\mathcal{T}$

Let us now consider how to avoid the construction of $\mathbb{T}$ and characterize the size of the reduced ordered decision diagram, $\mathcal{T}$. We begin by assuming that the underlying dynamics is well-approximated in the random-bit model.

**Definition 8** *For $q \in \mathbb{N}$, let $\mathbf{c} \sim \{0,1\}^q$ denote the random variable representing the result of flipping $q \in \mathbb{N}$ fair coins. We say a probabilistic automata $M = (S, s_0, A, \delta)$ is $(\epsilon, q)$ approximated in the random bit model if there exists a mapping,*

$$\hat{\delta} : S \times A \times \{0,1\}^q \to S \tag{27}$$

*such that for all $s, a, s' \in S \times A \times S$:*

$$\left| \delta(s, a, s') - \Pr_{\mathbf{c} \sim \{0,1\}^q}\left(\hat{\delta}(s, a, c) = s'\right) \right| \leq \epsilon. \tag{28}$$

For example, in our gridworld example (Fig 2a), if $\mathbf{c} \in \{0,1\}^3$, elements of $s$ are interpreted as pairs in $\mathbb{R}^2$, and the right/down actions are interpreted as the addition of the unit vectors $(1,0)$ and $(0,1)$ then,

$$\hat{\delta}(s, a, \mathbf{c}) = \begin{cases} s & \text{if } \max_i[(s+a)_i] > 1 \\ s + (0,1) & \text{else if } \mathbf{c} = 0 \\ s + a & \text{otherwise} \end{cases}, \tag{29}$$

As can be easily confirmed, (29) satisfies (28) with $\epsilon = 0$. In the sequel, we shall take access to $\hat{\hat{\delta}}$ as given[3]. Further, to simplify exposition, until Sec 5.1, we shall additionally require that the number of actions, $|A|$, be a power of 2. This assumption implies that $A$ can be encoded using exactly $\log_2(|A|)$ bits.

Under the above two assumptions, the key observation is to recognize that $\mathbb{T}$ (and thus $\mathcal{T}$) can be viewed as a Boolean predicate over an alternating sequence of action bit strings and coin flip outcomes determining if the task specification is satisfied, i.e.,

$$\mathbb{T} : \{0,1\}^n \to \{0,1\}, \tag{30}$$

where $n \overset{\text{def}}{=} \tau{\cdot}\log_2(|A{\times}A_e|) = \tau{\cdot}(q{+}\log_2(|A|))$. That is to say, the resulting decision diagram can be re-encoded as a reduced ordered **binary** decision diagram [3].

**Definition 9** *A* ***reduced ordered binary decision diagram*** *(BDD), is a representation of a Boolean predicate $h(x_1, x_2, \ldots, x_n)$ as a reduced ordered (deterministic) decision diagram, where each decision corresponds to testing a bit $x_i \in \{0,1\}$. We denote the BDD encoding of $\mathcal{T}$ as $\mathcal{B}$.*

Binary decision diagrams are well developed both in a theoretical and practical sense. Before exploring these benefits, we first note that this change has introduced an additional problem. First, note that in $\mathcal{B}$, decision and chance nodes from $\mathbb{T}$ are now encoded as sequences of decision and chance nodes. For example, if $a \in A$ is encoded by the 4-length bit sequence $b_1 b_2 b_3 b_4$, then four decisions are made by the agent before selecting an action. Notice however that the original semantics are preserved due to associativity of the softmax and $\mathbb{E}$ operators. In particular, recall that by definition,

$$\begin{aligned}\text{softmax}(\alpha_1, \ldots, \alpha_4) &= \ln(\sum_{i=1}^{4} e^{\alpha_i}) = \ln(e^{\ln(e^{\alpha_1}+e^{\alpha_2})} + e^{\ln(e^{\alpha_3}+e^{\alpha_4})}) \\ &\overset{\text{def}}{=} \text{softmax}(\text{softmax}(\alpha_1, \alpha_2), \text{softmax}(\alpha_3, \alpha_4))\end{aligned} \tag{31}$$

and thus the semantics of the sequence decision nodes is equivalent to the decision node in $\mathbb{T}$. Similarly, recall that the coin flips are fair, and thus expectations are computed via $\text{avg}(\alpha_1, \ldots, \alpha_n) = {}^1\!/\!{}_n(\sum_{i=1}^{n} \alpha_i)$. Therefore, averaging over two sequential coin flips yields,

$$\begin{aligned}\text{avg}(\alpha_1, \ldots, \alpha_4) &\overset{\text{def}}{=} \frac{1}{4}\sum_{i=1}^{4} \alpha_i = \frac{1}{2}(\frac{1}{2}(\alpha_1 + \alpha_2) + \frac{1}{2}(\alpha_3 + \alpha_4)) \\ &\overset{\text{def}}{=} \text{avg}(\text{avg}(\alpha_1, \alpha_2), \text{avg}(\alpha_3, \alpha_4))\end{aligned} \tag{32}$$

which by assumption (28), is the same as applying $\mathbb{E}$ on the original chance node. Finally, note that skipping over decisions needs to be adjusted slightly to account for sequences of decisions. Recall that via (26), the corresponding change in value,

[3] See [31] for an explanation on systematically deriving such encodings.

$\Delta$, is a function of initial value, $\alpha$, and the number of agent actions skipped, i.e., $|A|^n$ for $n$ skipped decision nodes. Thus, in the BDD, since each decision node has two actions, skipping $k$ decision bits corresponds to skipping $2^k$ actions. Thus, if $k$ decision bits are skipped over in the BDD, the change in value, $\Delta$, becomes,

$$\Delta(k, \alpha) = \alpha + k \ln(2). \tag{33}$$

Further, note that $\Delta$ can be computed in constant time while traversing the BDD. Thus, the dynamic programming scheme is linear in the size of $\mathcal{B}$.

**4.1 Size of $\mathcal{B}$:** Next we return to the question of how big the compressed decision diagram can actually be. To this aim, we cite the following (conservative) bound on the size of an BDD given an encoding of the corresponding Boolean predicate in the linear model computation illustrated in Fig 6 (for more details, we refer the reader to [16]).

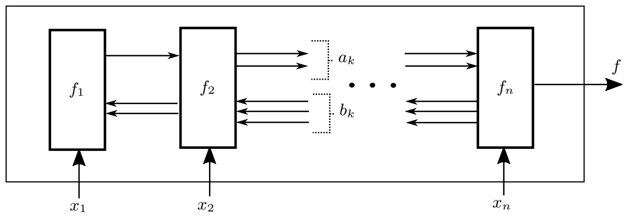


Fig. 6: Generic network of Boolean modules for which Theorem 1 holds.

In particular, consider an arbitrary Boolean predicate

$$f : \{0,1\}^n \to \{0,1\} \tag{34}$$

and a sequential arrangement of $n$ Boolean modules, $f_1, f_2, \ldots, f_n$ where each $f_i$ has shape:

$$f_i : \{0,1\} \times \{0,1\}^{a_{i-1}} \times \{0,1\}^{b_i} \to \{0,1\}^{a_i} \times \{0,1\}^{b_{i-1}}, \tag{35}$$

and takes as input $x_i$ as well as $a_{i-1}$ outputs of its left neighbor and $b_i$ outputs of the right neighbor ($b_0 = 0, a_n = 1$). Further, assume that this arrangement is well defined, e.g. for each assignment to $x_1, \ldots, x_n$ there exists a unique way to set each of the inter-module wires. We say these modules compute $f$ if the final output is equal to $f(x_1, \ldots, x_n)$.

**Theorem 1** *If $f$ can be computed by a linear arrangement of such modules, ordered $x_1, x_2, \ldots, x_n$, then the size, $S \in \mathbb{N}$, of its BDD (in the same order), is upper bounded [3] by:*

$$S \leq \sum_{k=1}^{n} 2^{a_k \cdot \left(2^{b_k}\right)}. \tag{36}$$

To apply this bound to our problem, recall that $\mathcal{B}$ computes a Boolean function where the decisions are temporally ordered and alternate between sequences of agent and environment decisions. Next, observe that because the traces are bounded (and all finite sets are regular), there exists a finite state machine which can monitor the satisfaction of the specification.

*Remark 5.* In the worst case, the monitor could be the unrolled decision tree, $\mathbb{T}$. This monitor would have exponential number of states. In practice, the composition of the dynamics and the monitor is expected to be much smaller.

Further, note that because this composed system is causal, no backward wires are needed, e.g., $\forall k \, . \, b_k = 0$. In particular, observe that because the composition of the dynamics and the monitor is Markovian, the entire system can be uniquely described using the monitor/dynamics state and agent/environment action (see Fig. 7). This description can be encoded in $\log_2(2^q|A \times S \times S_\varphi|)$ bits, where $q$ denotes the number of coin flips tossed by the environment and $S_\varphi$ denotes the monitor state. Therefore, $a_k$ is upper bounded by $\log_2(2^q|A \times S \times S_\varphi|)$. Combined with (36) this results in the following bound on the size of $\mathcal{B}$.

**Corollary 1** *Let $M = (S, s_0, A, \delta)$ be a probabilistic automaton whose probabilistic transitions can be approximated using $q$ coin flips and let $\varphi$ be a specification defined for horizon $\tau$ and monitored by a finite automaton with states $S_\varphi$. The corresponding BDD, $\mathcal{B}$, has size bounded by:*

$$|\mathcal{B}| \leq \overbrace{\tau \cdot \big(\log(|A|) + q\big)}^{\#\ inputs} \cdot \overbrace{\big(2^q|A \times S \times S_\varphi|\big)}^{bound\ on\ 2^{a_k}} \qquad (37)$$

Notice that the above argument implies that as the episode length grows, $|\mathcal{B}|$ grows linearly in the horizon/states and quasi-linearly in the agent/environment actions!

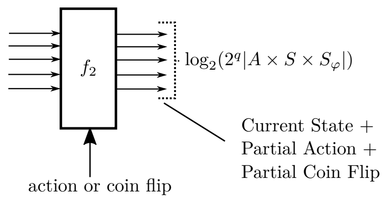

$f_2$
$\log_2(2^q|A \times S \times S_\varphi|)$
Current State + Partial Action + Partial Coin Flip
action or coin flip

Fig. 7: Generic module in linear model of computation for $\mathcal{B}$. Note that backward edges are not required.

*Remark 6.* Note that this bound actually holds for the minimal representation of the composed dynamics/monitor (even if it's unknown a-prori!). For example, if the property is *true*, the BDD requires only one state (always evaluate true). This also illustrates that the above bound is often very conservative. In particular, note that for $\varphi = true$, $|\mathcal{B}| = 1$ , independent of the horizon or dynamics. However, the above bound will always be linear in $\tau$. In general, the size of the BDD will depend on the particular symmetries compressed.

*Remark 7.* With hindsight, corollary 1 is not too surprising. In particular, if the monitor is known, then one could explicitly compose the dynamics MDP with the monitor, with the resulting MDP having at most $|S \times S_\varphi|$ states. If one then includes the time step in the state, one could perform the soft-Bellman Backup directly on this automaton. In this composed automaton each (action, state) pair would need to be recorded. Thus, one would expect $O(|S \times S_\varphi \times A|)$ space to be used. In practice, this explicit representation is much bigger than $\mathcal{B}$ due to the BDDs ability to skip over time steps and automatically compress symmetries.

**4.2 Constructing $\mathcal{B}$:** One of the biggest benefits of the BDD representation of a Boolean function is the ability to build BDDs from a Boolean combinations of other BDDs. Namely, given two BDDs with $n$ and $m$ nodes respectively, it is well known that the conjunction or disjunction of the BDDs has at most $n \cdot m$ nodes. Thus, in practice, if the combined BDD's remain relatively small, Boolean combinations remain efficient to compute and one does not construct the full binary decision tree! Further, note that BDDs support function composition. Namely, given predicates $f(x_1, \ldots, x_n)$ and $n$ predicates $g_i(y_1, \ldots, y_k)$ the function

$$f\Bigg(g_1(y_1, \ldots, y_k), \ldots, g_n(y_1, \ldots, y_k)\Bigg) \tag{38}$$

can be computed in time [16]:

$$O(n \cdot |B_f|^2 \cdot \max_i |B_{g_i}|), \tag{39}$$

where $B_f$ is the BDD for $f$ and $B_{g_i}$ are the BDDs for $g_i$. Now, suppose $\hat{\delta}_1, \ldots \hat{\delta}_{\log(|S|)}$ are Boolean predicates such that:

$$\hat{\delta}(\mathbf{s}, \mathbf{a}, \mathbf{c}) = (\hat{\delta}_1(\mathbf{s}, \mathbf{a}, \mathbf{c}), \ldots, \hat{\delta}_{\log(|S|)}(\mathbf{s}, \mathbf{a}, \mathbf{c})). \tag{40}$$

Thm 1 and an argument similar to that for Corr 1 imply then that constructing $\mathcal{B}$, using repeated composition, takes time bounded by a low degree polynomial in $|A \times S \times S_\varphi|$ and the horizon. Moreover, the space complexity before and after composition are bounded by Corr 1.

**4.3 Evaluating Demonstrations:** Next let us return to the question of how to evaluate the likelihood of a concrete demonstration in our compressed BDD. The key problem is that the BDD can only evaluate (binary) sequences of actions/coin flips, where as demonstrations are given as sequences of action/state pairs. That is, we need to algorithmically perform the following transformation.

$$s_0\mathbf{a}_0\mathbf{s}_1 \ldots \mathbf{a}_n\mathbf{s}_{n+1} \mapsto \mathbf{a}_1\mathbf{c}_1 \ldots \mathbf{a}_n\mathbf{c}_n \tag{41}$$

Given the random bit model assumption, this transformation can be rewritten as a series of Boolean Satisifability problems:

$$\exists\ \mathbf{c}_i\ .\ \hat{\delta}(\mathbf{s}_i, \mathbf{a}_i, \mathbf{c}_i) = \mathbf{s}_{i+1} \tag{42}$$

While potentially intimidating, in practice such problems are quite simple for modern SAT solvers, particularly if the number of coin flips used is small. Furthermore, many systems are translation invariant. In such systems, the results of a single query (42), can be reused on other queries. For example, in (29), $\mathbf{c} = \mathbf{0}$ always results in the agent moving to the right. Nevertheless, in general, if $q$ coin flips are used, encoding all the demonstrations takes at most $O(|X| \cdot \tau \cdot 2^q)$, in the worst case.

**4.4 Run-time analysis:** We are finally ready to provide a run-time analysis for our new inference algorithm. The high-level likelihood estimation procedure is described in Fig 8. First, the user specifies a dynamical system and a (multi-) set of demonstrations. Then, using a user-defined mechanism, a candidate task specification is selected. The system then creates a compressed representation of the composition of the dynamical system with the task specification. Then, in parallel, the maximum causal entropy policy is estimated and the demonstrations are themselves encoded as bit-vectors. Finally, the likelihood of generating the encoded demonstrations is computed.

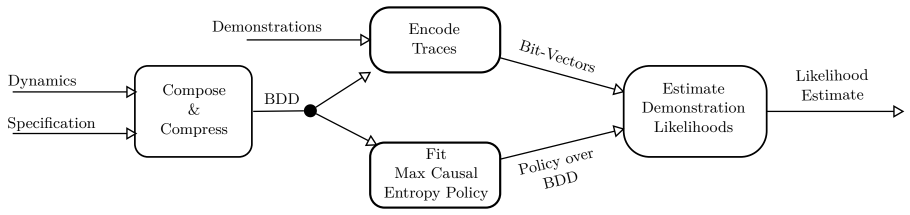


Fig. 8: High level likelihood estimation procedure described in this paper.

There are three computational bottlenecks in the compressed scheme. First, given a candidate specification, $\varphi$, one needs to construct $\mathcal{B}$. As argued in Sec 4.2, this takes time at most polynomial in the horizon, monitoring automata size, and MDP size (in the random-bit model). Second is the process of computing $Q$ and $V$ values by tuning the rationality coefficient to match a particular satisfaction probability. Just as with the naïve run-time (23), this process takes time linear in the size of $|\mathcal{B}|$ and logarithmic in the inverse tolerance $1/\epsilon$. Further, using Corr 1, we know that $|\mathcal{B}|$ is at most linear in horizon and quasi-linear in the MDP size. Thus, the policy computation takes time polynomial in the MDP size and logarithmic in the inverse tolerance. Finally, as before, evaluating the likelihoods takes time linear in the number of demonstrations and the horizon. However, we now require an additional step of finding coin-flips which are consistent with the demonstrations. Thus, the compressed run-time is bounded by:

$$O\Bigg(\Big(\underbrace{|X|}_{\#\text{Demos}} \cdot \overbrace{\log\left(\epsilon^{-1}\right)}^{\text{Feature Matching}}\Big) \cdot \text{POLY}\Big(\overbrace{\tau,}^{\text{Horizon}} \underbrace{|S|, |S_{|\varphi|}|, |A|,}_{\text{Composed MDP size}} \overbrace{2^q}^{\#\text{Coin Flip Outcomes}}\Big)\Bigg) \tag{43}$$

*Remark 8.* In practice, this analysis is fairly conservative since BDD composition is often fast, the bound given by Corr 1 is loose, and the SAT queries under-consideration are often trivial.

## 5 Additional Model Refinements

**5.1 Conditioning on Valid Actions:** So far, we have assumed that the number of actions is a power of 2. Functionally, this assumption makes it so each assignment to the action decision bits corresponds to a valid action. Of course, general MDPs have non-power of 2 action sets, and so it behooves us to adapt our method for such settings. The simplest way to do so is to use a 3-terminal Binary Decision Diagram. In particular, while each decision is still Boolean, there has now three possible types of leaves, 0, 1, and $\bot$. In the adapted algorithm, edges leading to $\bot$ are simply ignored, as they semantically correspond to invalid assignments to action or coin flip bits. A similar analysis can be done using these three valued decision diagrams, and as with BDDs, there exist efficient implementations of multi-terminal BDDs.

*Remark 9.* This generalization also opens up the possibility of state dependent action sets, where $A$ is now the union of all possible actions, e.g, disable the action for moving to the right when the agent is on the right edge of the grid.

**5.2 Choice of binary co-domain:** One might wonder how sensitive this formulation is to the choice of $R(\xi) = \theta \cdot \varphi(\xi)$. In particular, how does changing the co-domain of $\varphi$ from $\{0, 1\}$ to any other real values, i.e.,

$$\varphi' : (A \times S)^\tau \to \{a, b\},$$

change the likelihood estimates in our maximum causal entropy model. We briefly remark that, subject to some mild technical assumptions, almost any two real values could be used for $\varphi$'s co-domain. Namely, observe that unless both $a$ and $b$ are zero, the expected satisfaction probability, $p$, is in one-to-one correspondence with the expected value of $\varphi'$, i.e.,

$$\mathbb{E}[\varphi'] = a \cdot p + b \cdot (1 - p).$$

Thus, if a policy is feature matching for $\varphi$, it must be feature matching for $\varphi'$ (and vice-versa). Therefore, the space of consistent policies is invariant under such transformations. Finally, because the space of policies is unchanged, the maximum causal entropy policies must remain unchanged. In practice, we prefer the use of $\{0, 1\}$ as the co-domain for $\varphi$ since it often simplifies many calculations.

**5.3 Variable Episode Lengths (with discounting):** As earlier promised, we shall now discuss how to extend our model to include variable length episodes. For simplicity, we shall limit our discussion to the setting where at each time step, the probability that the episode will end is $\gamma \in (0, 1]$. As we previously discussed, this can be modeled by introducing a sink state, \$, representing the end of an episode (4). In the random bit model, this simply adds a few additional enviroment coin flips, corresponding to the enviroments new transitions to the sink state.

*Remark 10.* Note that when unrolled, once the end of episode transition happens, all decisions are assumed inconsequential w.r.t $\varphi$. Thus, all subsequent decisions will be compressed by in the BDD, $\mathcal{B}$.

Finally, observe that the probability that the episode ending increases exponentially, implying that the planning horizon need not be too big, i.e., the probability that the episode has not ended by timestep, $\tau \in \mathbb{N}$, is: $(1 - \gamma)^\tau$. Thus, letting $\tau = \lceil \ln(\epsilon/1-\gamma) \rceil$ ensures that with probability at least $1 - \epsilon$ the episode has ended.

## 6 Experiment

Below we report empirical results that provide evidence that our proposed technique is robust to demonstration errors and that the produced BDDs are smaller than a naïve dynamic programming scheme. To this end, we created a reference implementation [29] in Python. BDD and SAT solving capabilities are provided via dd [21] and pySAT [12] respectively. To encode the task specifications and the random-bit model MDP, we leveraged the py-aiger ecosystem [28] which includes libraries for modeling Markov Decision Processes and encoding Past Tense Temporal Logic as sequential circuits.

**Problem:** Consider a gridworld where an agent can attempt to move up, down, left, or right; however, with probability 1/32, the agent slips and moves left. Further, suppose a demonstrator has provided the six unlabeled demonstrations shown in Fig. 9 for the task: "Within 10 time steps, touch a yellow (recharge) tile while avoiding red (lava) tiles. Additionally, if a blue (water) tile is stepped on, the agent must step on a brown (drying) tile before going to a yellow (recharge) tile." All of the solid paths satisfy the task. The dotted path fails because the agent keeps slipping left and thus cannot dry off by $t = 10$. Note that due to slipping, all the demonstrations that did not enter the water are sub-optimal.

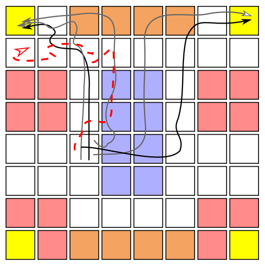


Fig. 9

| Spec | Policy Size (#nodes) | ROBDD build time | Relative Log Likelihood (compared to True) |
|---|---|---|---|
| true | 1 | 0.48s | 0 |
| $\varphi_1$ = Avoid lava | 1797 | 1.5s | -22 |
| $\varphi_2$ = Eventually Recharge | 1628 | 1.2s | 5 |
| $\varphi_3$ = Don't recharge while wet | 850 | 1.6s | -10 |
| $\varphi_4 \overset{\text{def}}{=} \varphi_1 \wedge \varphi_2$ | 523 | 1.9s | 4 |
| $\varphi_5 \overset{\text{def}}{=} \varphi_1 \wedge \varphi_3$ | 1913 | 1.5s | -2 |
| $\varphi_6 \overset{\text{def}}{=} \varphi_2 \wedge \varphi_3$ | 1842 | 2s | 15 |
| $\varphi^* \overset{\text{def}}{=} \varphi_1 \wedge \varphi_2 \wedge \varphi_3$ | 577 | 1.6s | 27 |

**Results:** For a small collection of specifications, we have computed the size of the BDD, the time it took to construct the BDD, and the *relative* log likelihoods of the demonstrations[4],

$$\text{RelativeLogLikelihood}(\varphi) \overset{\text{def}}{=} \ln\left(\frac{\Pr(\text{demos} \mid \varphi)}{\Pr(\text{demos} \mid \text{true})}\right), \tag{44}$$

where each maximum entropy policy was fit to match the corresponding specification's empirical satisfaction probability. We remark that the computed BDDs are small compared to other straw-man approaches. For example, an explicit construction of the product of the monitor, dynamics, and the current time step would require space given by:

$$\tau \cdot |S| \cdot |A| \cdot |S_\varphi| = (10 \cdot 8 \cdot 8 \cdot 4) \cdot |S_\varphi| = 2560 \cdot |S_\varphi| \tag{45}$$

[4] The maximum entropy policy for $\varphi = \text{true}$ applies actions uniformly at random.

The resulting BDDs are much smaller than (45) and the naïve unrolled decision tree. We note that the likelihoods appear to (qualitatively) match expectations. For example, **despite** an unlabeled negative example, the demonstrated task, $\varphi^*$, is the most likely specification. Moreover, under the second most likely specification, which omits the avoid lava constraint, the sub-optimal traces that do not enter the water appear more attractive.

Finally, to emphasize the need for our causal extension, we compute the likelihoods of $\varphi^*, \varphi_1, \varphi_2$ for our opening example (Fig. 1) using both our causal model and the prior non-causal model [30]. Concretely, we take $\tau = 15$, a slip probability of 1/32, and fix the expected satisfaction probability to 0.9. The trace shown in Fig. 1 acts as the sole (failed) demonstration for $\varphi^*$. As desired, our causal extension assigned more than 3 times the relative likelihood to $\varphi^*$ compared to $\varphi_1$, $\varphi_2$, and *true*. By contrast, the non-causal model assigns relative log likelihoods $(-2.83, -3.16, -3.17)$ for $(\varphi_1, \varphi_2, \varphi^*)$. This implies that (i) $\varphi^*$ is the least likely specification and (ii) each specification is less likely than *true*!

## 7 Conclusion and Future Work

Motivated by the problem of learning specifications from demonstrations, we have adapted the principle of maximum causal entropy to provide a posterior probability to a candidate task specification given a multi-set of demonstrations. Further, to exploit the structure of task specifications, we proposed an algorithm that computes this likelihood by first encoding the unrolled Markov Decision Process as a reduced ordered binary decision diagram (BDD). As illustrated on a few toy examples, BDDs are often much smaller than the unrolled Markov Decision Process and thus could enable efficient computation of maximum causal entropy likelihoods, at least for well behaved dynamics and specifications.

Nevertheless, two major questions remain unaddressed by this work. First is the question of how to select which specifications to compute likelihoods for. For example, is there a way to systematically mutate a specification to make it more likely and/or is it possible to systematically reuse computations for previously evaluated specifications to propose new specifications.

Second is how to set prior probabilities. Although we have largely ignored this question, we view the problem of setting good prior probabilities as essential to avoid over fitting and/or making this technique require only one or two demonstrations. However, we note that prior probabilities can make inference arbitrarily more difficult since any structure useful for optimization imposed by our likelihood estimate can be overpowered.

Finally, additional future work includes extending the formalism to infinite horizon specifications, continuous dynamics, and characterizing the optimal set of teacher demonstrations.

**Acknowledgments**: We would like to thank the anonymous referees as well as Daniel Fremont, Ben Caulfield, Marissa Ramirez de Chanlatte, Gil Lederman, Dexter Scobee, and Hazem Torfah for their useful suggestions and feedback. This work was supported in part by NSF grants 1545126 (VeHICaL) and 1837132, the DARPA BRASS program under agreement number FA8750-16-C0043, the DARPA Assured Autonomy program, Toyota under the iCyPhy center, and Berkeley Deep Drive.

## References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Information and computation **75**(2), 87–106 (1987)
2. Bellman, R.E., et al.: Dynamic programming, ser. In: Rand Corporation research study. Princeton University Press (1957)
3. Bryant, R.E.: Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys (CSUR) **24** (1992)
4. Farwer, B.: $\omega$-automata. In: Automata logics, and infinite games, pp. 3–21. Springer (2002)
5. Finn, C., Levine, S., Abbeel, P.: Guided cost learning: Deep inverse optimal control via policy optimization. In: International Conference on Machine Learning. pp. 49–58 (2016)
6. Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., Levine, S.: Composable deep reinforcement learning for robotic manipulation. arXiv preprint arXiv:1803.06773 (2018)
7. Heule, M.J., Verwer, S.: Exact DFA identification using SAT solvers. In: International Colloquium on Grammatical Inference. pp. 66–79. Springer (2010)
8. De la Higuera, C.: Grammatical inference: learning automata and grammars. Cambridge University Press (2010)
9. Hoey, J., St-Aubin, R., Hu, A., Boutilier, C.: Spudd: Stochastic planning using decision diagrams. In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. pp. 279–288. Morgan Kaufmann Publishers Inc. (1999)
10. Holtzen, S., Millstein, T.D., den Broeck, G.V.: Symbolic exact inference for discrete probabilistic programs. CoRR **abs/1904.02079** (2019), http://arxiv.org/abs/1904.02079
11. Icarte, R.T., Klassen, T., Valenzano, R., McIlraith, S.: Using reward machines for high-level task specification and decomposition in reinforcement learning. In: International Conference on Machine Learning. pp. 2112–2121 (2018)
12. Ignatiev, A., Morgado, A., Marques-Silva, J.: PySAT: A Python toolkit for prototyping with SAT oracles. In: SAT. pp. 428–437 (2018)
13. Jaynes, E.T.: Information theory and statistical mechanics. Physical review **106**(4), 620 (1957)
14. Jothimurugan, K., Alur, R., Bastani, O.: A composable specification language for reinforcement learning tasks. In: Advances in Neural Information Processing Systems. pp. 13021–13030 (2019)
15. Kasenberg, D., Scheutz, M.: Interpretable apprenticeship learning with temporal logic specifications. arXiv preprint arXiv:1710.10532 (2017)
16. Knuth, D.E.: The Art of Computer Programming: Vol. 4, No. 1: Bitwise Tricks and Techniques-Binary Decision Diagrams. Addison Wesley Professional (2009)
17. Kramer, G.: Directed information for channels with feedback. Hartung-Gorre (1998)
18. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd International Conference on Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 585–591. Springer (2011)
19. Levine, S.: Reinforcement learning and control as probabilistic inference: Tutorial and review. CoRR **abs/1805.00909** (2018), http://arxiv.org/abs/1805.00909
20. Levine, S., Popovic, Z., Koltun, V.: Nonlinear inverse reinforcement learning with gaussian processes. In: Advances in Neural Information Processing Systems 24 (2011)

21. Livingston, S.C.: Binary Decision Diagrams (BDDs) in pure Python and Cython wrappers of CUDD, Sylvan, and BuDDy
22. Ng, A.Y., Russell, S.J., et al.: Algorithms for inverse reinforcement learning. In: ICML. pp. 663–670 (2000)
23. Permuter, H.H., Kim, Y.H., Weissman, T.: On directed information and gambling. In: 2008 IEEE International Symposium on Information Theory. pp. 1403–1407. IEEE (2008)
24. Pineau, J., Gordon, G., Thrun, S., et al.: Point-based value iteration: An anytime algorithm for POMDPs. In: IJCAI. vol. 3, pp. 1025–1032 (2003)
25. Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. IJCAI (2007)
26. Todorov, E.: Linearly-solvable Markov decision problems. In: Advances in neural information processing systems. pp. 1369–1376 (2007)
27. Todorov, E.: General duality between optimal control and estimation. In: Decision and Control, 2008. CDC 2008. 47th IEEE Conference on. pp. 4286–4292. IEEE (2008)
28. Vazquez-Chanlatte, M.: mvcisback/py-aiger (Aug 2018). https://doi.org/10.5281/zenodo.1326224, https://doi.org/10.5281/zenodo.1326224
29. Vazquez-Chanlatte, M.: mce-spec-inference. https://github.com/mvcisback/mce-spec-inference/ (2020)
30. Vazquez-Chanlatte, M., Jha, S., Tiwari, A., Ho, M.K., Seshia, S.: Learning task specifications from demonstrations. In: Advances in Neural Information Processing Systems 31, pp. 5368–5378 (2018)
31. Vazquez-Chanlatte, M., Rabe, M.N., Seshia, S.A.: A model counter's guide to probabilistic systems. arXiv preprint arXiv:1903.09354 (2019)
32. Ziebart, B.D., Bagnell, J.A., Dey, A.K.: Modeling interaction via the principle of maximum causal entropy (2010)
33. Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: AAAI. vol. 8, pp. 1433–1438. Chicago, IL, USA (2008)