Demonstration Informed Specification Search

Marcell Vazquez-Chanlatte[1], Ameesh Shah[1], Gil Lederman[1], and Sanjit A. Seshia[1]

[1]University of California, Berkeley

Abstract. This paper considers the problem of learning history dependent task specifications, e.g. automata and temporal logic, from expert demonstrations. Unfortunately, the (countably infinite) number of tasks under consideration combined with an a-priori ignorance of what historical features are needed to encode the demonstrated task makes existing approaches to learning tasks from demonstrations inapplicable. To address this deficit, we propose *Demonstration Informed Specification Search (DISS)*: a family of algorithms parameterized by *black box* access to (i) a maximum entropy planner and (ii) an algorithm for identifying concepts, e.g., automata, from labeled examples. DISS works by alternating between (i) conjecturing labeled examples to make the demonstrations less surprising and (ii) sampling concepts consistent with the current labeled examples. In the context of tasks described by deterministic finite automata, we provide a concrete implementation of DISS that efficiently combines partial knowledge of the task and a single expert demonstration to identify the full task specification.

1 Introduction

Expert demonstrations provide an accessible and expressive means to informally specify a task, particularly in the context of human robot interaction. In this work, we study the problem of inferring, from demonstrations, tasks represented by formal *task specifications*, e.g., automata and temporal logic. The study of task specifications is motivated by their ability to (i) encode historical dependencies, (ii) incrementally refine the task via composition, and (iii) be semantically robust to changes in the workspace, e.g., changing the transition probabilities does not change the set paths that satisfy the specification [14].

1.1 Related Work The problem of learning objectives by observing an expert has a rich and well developed literature dating back to early work on Inverse Optimal Control [6] and more recently via Inverse Reinforcement Learning (IRL) [9]. In IRL, an expert demonstrator optimizes an *unknown* reward function by acting in a stochastic environment. The goal of IRL is to find a reward function that explains the agent's behavior. A fruitful approach has been to cast IRL as a Bayesian inference problem to predict the most probable reward function [10]. To make this inference robust to demonstration/modeling noise, one commonly appeals to the principle of maximum causal entropy [5, 19, 18]. Intuitively, this

results in a forecasting model that is no more committed to any given action than the data requires.

While powerful, traditional IRL provides no principled mechanism for composing the resulting reward artifacts and requires the relevant historical features (memory) to be a-priori known. To address this deficit, recent works have proposed learning Boolean task specifications, e.g. logic or automata, which admit well defined compositions and explicitly encode temporal constraints. The development of this literature mirrors the historical path taken in reward based research, with works adapting optimal control [7, 2], Bayesian [11, 17], and maximum entropy approaches [14, 15] IRL approaches.

A key difficulty for the task specification inference from demonstrations literature is how to search an intractably large (often infinite) concept class. In particular, and in contrast to the reward setting, the discrete nature of automata and logic, combined with the assumed *a-priori* ignorance of the relevant memory required to describe the task, makes existing gradient based approaches either intractable or inapplicable. Instead, current literature either enumerates concepts [14, 2, 11, 17] or hill climbs via simple probabilistic mutations [7, 1].

The major contribution of this paper is to systematically reduce the problem of learning from demonstrations into a series of supervised task specification identification problems, e.g., finding a small Deterministic Finite Automta (DFA) that is consistent with a set of positive and negative strings [3][4]. The result is a principled way to sample tasks given a candidate task. This insight is integrated into a variant of simulated annealing [12] for *guided* hill climbing.

1.2 Motivating Example Consider an agent operating in the 8x8 grid world as shown in Fig 1. The agent can attempt to move up, down, left, or right. With probability $1/32$, wind will push the agent down, regardless of the agent's action. The black path is the *prefix* of an episode, in which the agent attempts to move left, slips into the blue tile (■), visits a brown tile (■), and then proceeds downward. Given the black demonstration, call ξ_b, and the *prior* knowledge that the agent's task implies that it will (i) avoid red tiles (■) and (ii) try to reach a yellow tile (■), what task, as a DFA, explains the agent's behavior?

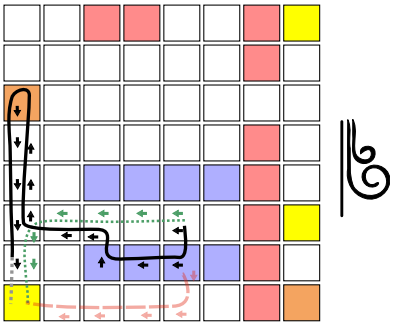


Fig. 1

Upon inspecting the demonstration, one might be surprised that the agent goes out of its way to visit ■. For example, why would the agent not take the red dashed path directly to ■? One might conjecture that the agent's true task requires visiting ■ after visiting ■. Similarly, one may notice that the demonstration is less surprising if the dotted extension of ξ_b ending in ■ is a positive example of the task. Finally, appealing to Occam's razor, one might search for a simple DFA consistent with the conjectured labels. In this case, the demonstrated task is in fact one of the "simplest" DFAs, shown on the right in

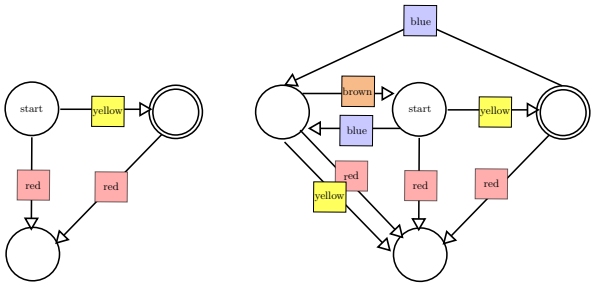


Fig. 2: DFAs with stuttering semantics, i.e., if a transition is not provided, a self loop is assumed. The accepting states are marked with a concentric circle and the initial state is labeled start. A sequence of inputs, e.g. colored, is accepted if the final state is accepting. The left DFA encodes: "Avoid ■ and eventually reach ■". The right DFA adds to the left DFA the rule: "Visitations of ■ and ■ must be separated by a visit to ■."

Fig 2. As we shall later see, DISS systematizes this line of reasoning and is able to learn explanatory DFAs, even without the prior task knowledge used in this motivating example!

1.3 Contributions and Algorithm Overview This work contributes a family of approximate algorithms called *Demonstration Informed Specification Search* (DISS) for the task specification inference from demonstrations problem.

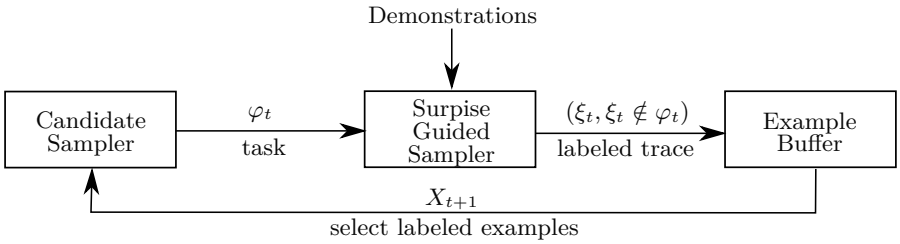


Fig. 3: Overview of Demonstration Informed Specification Search

DISS assumes (i) access to a multi-set of expert demonstrations: $\xi_1^*, \ldots \xi_m^*$, (ii) *black box* access to an identification algorithm, $\mathcal{I}$, that maps positively/negatively labeled paths to a distribution over concepts and (iii) *black box* access to a planner that estimates the probability of a path given a candidate task (see Sec 3). DISS operates by cycling between three components (shown in Fig 3):

1. **Candidate Sampler**: A candidate task, φ, is sampled from $\mathcal{I}(X)$, where X is a collection of labeled examples (initialized to the empty set).

2. **Surprise Guided Sampler**: The planner is used to find a path, where toggling whether the path is a positive/negative example of the task makes the expert demonstrations less surprising.
3. **Example Buffer**: Given previously seen data, the example buffer yields a set of positive and negative example paths.

In the sequel, we will (i) formalize the problem statement, (ii) propose a agent model based on the extensive literature on maximum causal entropy agent models, (iii) discuss how to find surprising paths given our planner, and (iv) propose a variant of simulated annealing (implemented through the example buffer) to approximately solve our task inference from demonstration problem.

> The resulting algorithm is *agnostic* to the underlying concept class and dynamics. The overhead introduced by DISS is only proportional to the number of demonstrations. The rest of the complexity is relativized in terms of the maximum entropy planner and concept identifier.

Finally, we provide a concrete implementation of DISS [16]. An example identification algorithm for DFAs and a maximum entropy planner for simple gridworlds are also included. Using this implementation, we perform two experiments validating that DISS indeed enables efficiently searching for tasks that explain the demonstrations even in large/unstructured concept classes like DFAs given unlabeled and potentially incomplete demonstrations.

Remark 1. The choice of DFAs as the concept class for our experiments was motivated by three observations. First, DFAs explicitly encode memory, making the contribution of identifying relevant memory more clear. Next, to our knowledge, all other techniques for learning finite path properties from demonstrations focus on syntax defined concept classes. Thus learning DFAs is understudied in this context. Third, DFAs constitute a very large and mostly unstructured concept class. Therefore, DFAs facilitate studying the efficiency of DISS without introducing too much user defined inductive biases. By comparison, existing techniques for learning task specifications from demonstrations all use use syntactically defined logics each with their own inductive biases. Thus direct comparisons would conflate search efficiency with the inductive biases of the concept classes.

2 Preliminaries and Problem Statement

2.1 Dynamics Model We model the expert *demonstrator* as operating in a *Markov Decision Process* (MDP), $M = (S, A, s_0, \delta)$, where (i) S denotes a finite set of states, (ii) $A(s)$ denotes the finite set of actions available at state $s \in S$, (iii) s_0 is initial state, (iv) $\delta(s' \mid a, s)$ is the probability of transitioning from s to s' when applying action $a \in A(s)$. We will make two additional assumptions about M. First, we assume a unique (always reachable) sink state, i.e., $\delta(\$ \mid a, \$) = 1$, denoting "end of episode". Second, we shall assert the Luce choice axiom, which requires that each action, $a \in A(s)$, be *distinct*, i.e., no actions are interchangeable or redundant at a given state [8].

A *path*, ξ, is an alternating sequence of states and actions starting with s_0:

$$\xi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots. \tag{1}$$

Any path, ξ, can be (non-uniquely) decomposed into a *prefix*, ρ, concatenated with a *suffix*, σ, denoted $\xi = \rho \cdot \sigma$. We allow σ to be of length 0. The last state of ξ is denoted by $\text{last}(\xi)$. A path is *complete* if it contains \$ exactly once, and thus $\text{last}(\xi) = \$$. We denote by $\text{Path}_\$$ the set of all complete paths, and by Path the set of all prefixes of $\text{Path}_\$$, i.e., paths that contain \$ at most once.

2.2 Task Specifications Next, we develop the machinery to describe the set of paths that constitute performing a given task. In particular, a *task specification* or *task* is a subset of paths,

$$\varphi \subseteq \text{Path}_\$. \tag{2}$$

We refer to a collection of task specifications, Φ, as a *concept class*. Given a concept class, the function *size* is a complexity measure that maps tasks to a description length,

$$\text{size} : \Phi \to \mathbb{R}_{\geq 0}. \tag{3}$$

Example 1. We formalize the concept class of our motivating example. Let (i) $\Sigma = \{\blacksquare, \blacksquare, \blacksquare, \blacksquare, \square\}$ denote an alphabet, (ii) D_1 and D_2 denote the left and right DFA over Σ shown in Fig 2, and (iii) φ_1 and φ_2 denote the set of paths whose corresponding color sequence is accepted by D_1 and D_2 resp. Define Φ_{reg} as the set of all tasks represented by a DFA over Σ, i.e. regular languages, and define Φ'_{reg} to be the set of regular languages that imply (are subsets of) φ_1. Note that $\varphi_1, \varphi_2 \in \Phi'_{\text{reg}} \subsetneq \Phi_{\text{reg}}$. Next, let $|\varphi|$ denote the number of non-stuttering (not self loop) edges in the minimal DFA for φ's multi-graph. Finally, for Φ_{reg} define $\text{size}(\varphi) \stackrel{\text{def}}{=} |\varphi|$ and for Φ'_{reg} define $\text{size}'(\varphi) \stackrel{\text{def}}{=} |\varphi| - |\varphi_1|$. For example, $\text{size}(\varphi_2) = 6$, and $\text{size}'(\varphi_R) = 6 - 3 = 3$.

A *labeled example* is tuple, $x = (\xi, l)$, corresponding to a complete path and a binary label, $l \in \{0, 1\}$. A collection of labeled examples, $X = x_1, \dots, x_n$, is *consistent* with a task, φ, if:

$$\forall x_i = (\xi_i, l_i) \;.\; (\xi_i \in \varphi) \iff (l = 1). \tag{4}$$

An *identification algorithm*, $\mathcal{I}$, maps a collection of labeled examples, X to a distribution over consistent tasks in Φ or $\perp$ if no consistent task exists.

Example 2. Let ξ_b and ξ_r be the completed black and red paths shown in Fig 1 and define $X_{bg} = \{(\xi_b, 1), (\xi_r, 0)\}$. φ_2 is consistent with X_{bg} and φ_1 is not.

2.3 Policies and Demonstrations A (history dependent) *policy*, $\pi(a \mid \xi)$, is a distribution over actions, a, given a path, ξ, where $a \in A(\text{last}(\xi))$. A policy is (p, φ)-*competent* if:

$$\text{psat}_\varphi(\pi) \stackrel{\text{def}}{=} \Pr(\xi \in \varphi \mid \pi, M) = p \tag{5}$$

A *demonstration*, is a path, ξ^*, generated by a employing a policy π in an MDP M, $\xi \sim (\pi, M)$.

Task Inference from Demonstrations Problem (TIDP): Let a M, Φ, and P be a fixed MDP, concept class, and task prior, respectively. Further, let π^* be a (p^*, φ^*)-competent policy, π^*, where p^*, φ^*, and π^* are all unknown. Given a multi-set of i.i.d. demonstrations, $\xi_1^*, \dots \xi_m^* \sim (\pi^*, M)$, find:

$$\varphi \in \arg\max_{\psi \in \Phi} \Pr(\xi_1^*, \dots \xi_m^* \mid \psi, M) \cdot P(\psi \mid M). \tag{6}$$

Of course, by itself, the above formulation is ill-posed as $\Pr(\xi_1^*, \dots \xi_m^* \mid M, \varphi)$ is left undefined. What remains is to derive a suitable agent model and discuss how to manipulate likelihoods in this model.

3 Task Motivated Agents

In the sequel, we follow [15] and propose formalizing the above problem statement by first estimating p^* given a candidate task, φ, and then assigning a bias-minimizing belief of generating the demonstrations given φ.

3.1 Maximum Causal Entropy Policies We start by defining the causal entropy on arbitrary sequences of random variables. Let $\mathcal{X}_{1:i} \stackrel{\text{def}}{=} \mathcal{X}_1, \dots, \mathcal{X}_i$ and $\mathcal{Y}_{1:i} \stackrel{\text{def}}{=} \mathcal{Y}_1, \dots, \mathcal{Y}_i$ denote two sequences of random variables. The probability of $\mathcal{X}_{1:i}$ *causally conditioned* on $\mathcal{Y}_{1:i}$ is:

$$\Pr(\mathcal{X}_{1:i} \parallel \mathcal{Y}_{1:i}) \stackrel{\text{def}}{=} \prod_{j=1}^{i} \Pr(\mathcal{X}_j \mid \mathcal{X}_{1:j-1}, \mathcal{Y}_{1:j}). \tag{7}$$

The *causal entropy* of $\mathcal{X}_{1:i}$ given $\mathcal{Y}_{1:i}$ is then defined as,

$$H(\mathcal{X}_{1:i} \mid\mid \mathcal{Y}_{1:i}) \stackrel{\text{def}}{=} -\mathop{\mathbb{E}}_{\mathcal{X}_{1:i}, \mathcal{Y}_{1:i}} [\ln \Pr(\mathcal{X}_{1:i} \parallel \mathcal{Y}_{1:i})] \tag{8}$$

Using the chain rule, one can relate causal entropy to (non-causal) entropy, $H(\mathcal{X} \mid \mathcal{Y}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathcal{X}}[-\ln \Pr(\mathcal{X} \mid \mathcal{Y})]$ via $H(\mathcal{X}_{1:i} \mid\mid \mathcal{Y}_{1:i}) = \sum_{t=1}^{i} H(\mathcal{X}_t \mid \mathcal{Y}_{1:t}, \mathcal{X}_{1:t-1})$. This relation shows that: (1) Causal entropy is always lower bounded by non-causal entropy (and thus non-negative). (2) Causal entropy can be computed "backward in time". Intuitively, and contrary to non-causal entropy, causal entropy does *not* condition on variables that have not been revealed, e.g., on events in the future. This makes causal entropy particularly well suited for robust forecasting in *sequential* decision making problems, as the agents cannot observe the future [18].

We define the τ-length path causal entropy of a policy, π as:

$$H_\tau(\pi) \stackrel{\text{def}}{=} H(\mathcal{A}_{1:\tau} \parallel \mathcal{S}_{1:\tau}), \tag{9}$$

where we denote the sequences of action and state random variables, $\mathcal{A}_{1:\tau} \stackrel{\text{def}}{=} \mathcal{A}_1, \dots, \mathcal{A}_\tau$ and $\mathcal{S}_{1:\tau} \stackrel{\text{def}}{=} \mathcal{S}, \dots, \mathcal{S}_\tau$, generated by (π, M).

For MDPs, the unique policy, π_φ, that maximizes entropy subject to being (p, φ)-competent is given by the following smoothed Bellman-backup:

$$\begin{aligned} \ln \pi_\lambda(a \mid \xi) &\stackrel{\text{def}}{=} Q_\lambda(\xi \cdot a) - V_\lambda(\xi) \\ Q_\lambda(\xi \cdot a) &\stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{\xi'}\left[V_\lambda(\xi') \mid a, \xi\right] \\ V_\lambda(\xi) &\stackrel{\text{def}}{=} \begin{cases} \lambda \cdot [\xi \in \varphi] & \text{if } \xi \text{ is complete} \\ \log \sum_{a \in A(\text{last}(\xi))} \exp\Big(Q_\lambda(\xi \cdot a)\Big) & \text{otherwise} \end{cases} \end{aligned}, \tag{10}$$

where λ, called the *rationality*, is set such that $\text{psat}_\varphi(\pi_\lambda) = p$. When clear from context, we will often write V_φ, and Q_φ, or even just V and Q.

Remark 2. $\text{psat}_\varphi(\pi_\lambda)$ increases monotonically in λ and thus can be efficiently calculated to arbitrary precision using binary search.

Remark 3. The competency of the agent can be treated as a hyper-parameter or estimated empirically, e.g., $p_\varphi \approx 1/m \sum_{i=1}^{m} [\xi \in \varphi]$. The former is useful when given on a few demonstrations and the latter is useful when given a large number of demonstrations.

3.2 Explainability of a task The *surprise* (or self-information) of a demonstration is defined as:

$$h(\xi \mid \pi, M) \stackrel{\text{def}}{=} -\ln \Pr(\xi \mid \pi, M). \tag{11}$$

The surprise of a collection of demonstrations is the sum of the surprise of each demonstration: $h(\xi_1^*, \ldots \xi_m^* \mid \pi, M) \stackrel{\text{def}}{=} \sum_{i=1}^{m} h(\xi_i \mid \pi, M)$. Note that the likelihood of i.i.d., demonstrations from (π, M) is simply $\exp(-h(\xi_1^*, \ldots \xi_m^*))$. Given a *fixed* MDP, M, and a *fixed* collection of demonstrations, $\xi_1, \ldots, \xi_m$, we define the surprise of a task, φ, as:

$$h(\varphi) \stackrel{\text{def}}{=} h(\xi_1^*, \ldots \xi_m^* \mid \pi_\varphi, M) \tag{12}$$

Thus, solving a TIDP requires minimizing h and the negative log prior, which w.l.o.g. can be taken as $\text{size}(\varphi)$.

4 Manipulating Surprise

In the sequel, we seek to study how changing a task φ changes the corresponding surprise, $h(\varphi)$, and thus the likelihood of observing the demonstrations.

4.1 Prefix Tree Perspective To facilitate this, we will find it useful to re-frame paths as either *deviating* or *conforming* to the demonstrations. To start, denote by $\mathcal{T} = (N, E)$ the *prefix tree* (or trie) of the demonstrations, $\xi_1^*, \ldots \xi_m^*$, where N and E are the *nodes* and *edges* of $\mathcal{T}$, respectively. Each node

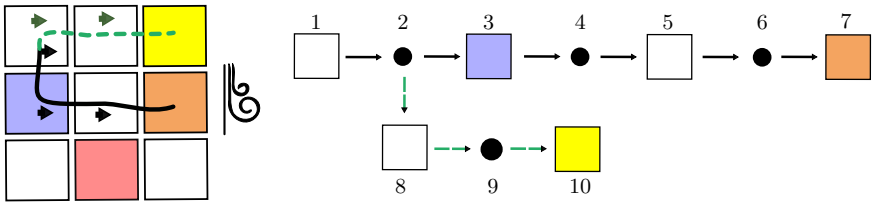


Fig. 4: Prefix tree with 12 nodes for the two paths shown on the left.

$i \in N$, corresponds to a prefix, ρ_i, of at least one of the demonstrations. An edge connects *parent*, i, to child, c, if ρ_c is the one action (or state) extension of ρ_i, i.e., $\rho_c = \rho_i\sigma$, where σ is a path of length 1. For each edge, $(i, j) \in E$, we define the *edge traversal count*, $\#_{(i,j)}$, as the number of times prefix ρ_j appears in the demonstrations. The set of unique demonstrations maps directly to the leaves of $\mathcal{T}$, where ρ_i is a demonstration for each leaf v. A node is said to be an *ego node* if it corresponds to selecting an action, and an *environment (env) node* otherwise.

We say a path, ξ, *conforms* to the demonstrations if there is a node i such that, $\rho_i = \xi$. A path *deviates* from the demonstrations if it is not conforming. The *pivot* of a deviating path, ξ, is the node corresponding to the longest prefix of ξ that conforms to the demonstrations. Note that it is possible to pivot at the leaves of the tree, i.e., the longest prefix is a demonstration.

Example 3. Example demonstrations and the correspondng prefix tree are illustrated in Fig 4. Note that it is possible to pivot at every node *except* node 2, since both possibilities (slipping/not slipping) appear in the demonstrations.

4.2 Proxy Surprise Next, observe that because weighted averaging and LSE are commutative, one can aggregate the values of a set of actions or set of states (environment actions) that deviate at pivot i. In particular let A_i and S_j denote the *conforming actions* and *conforming states* at an ego node i and an env node j, respectively. The *pivot value*, of a node i is defined as:

$$\mathbb{V}_i^{\varphi} \overset{\text{def}}{=} \begin{cases} \ln \sum_{a \notin A_i} \exp\Big(Q_\varphi(\rho_i \cdot a)\Big) & \text{if } i \text{ is an ego node,} \\ \mathbb{E}_s[V_\varphi(\rho_i \cdot s) \mid s \notin S_i, \rho_i, M] & \text{if } i \text{ is an env node,} \end{cases} \tag{13}$$

We shall denote by $\mathbb{V}^\varphi \in \mathbb{R}^N$ the node-indexed vector of pivot values associated with task φ under our maximum entropy agent model. Crucially, the pivot values **entirely determine** (see Fig 5) the values of the states and actions visited by the demonstrations via (10). Namely, let $\hat{Q}_k(\mathbb{V})$ and $\hat{V}_k(\mathbb{V})$ denote the derived action and state value at node k in the prefix tree, and let $\Pr(i \rightsquigarrow k \mid \mathbb{V})$ denote the probability of transitioning from node i to node k under the local maxEnt

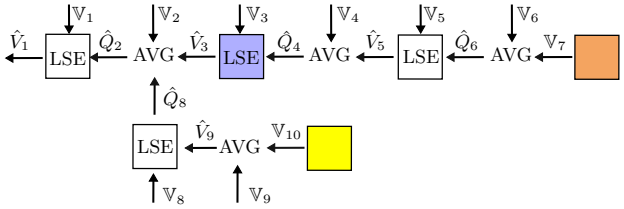


Fig. 5: Computation tree of $\hat{Q}$ and $\hat{V}$ values for each node of prefix tree given by soft Bellman backup (10) and node pivot values, $\mathbb{V}$.

policy for prefix tree corresponding to $\mathbb{V}$. The *proxy surprise*, $\hat{h}$, is given by:

$$\hat{h}(\mathbb{V}) \stackrel{\text{def}}{=} -\sum_{(i,j)\in E} \#_{(i,j)} \cdot \ln \Pr(i \rightsquigarrow j \mid \mathbb{V}),$$

$$\ln \Pr\Big(i \rightsquigarrow k \mid \mathbb{V}, (i,k) \in E\Big) = \begin{cases} \Pr(\rho_k \mid \rho_i, a, M) & \text{if } i \text{ is env,} \\ \hat{Q}_k(\mathbb{V}) - \hat{V}_i(\mathbb{V}) & \text{if } i \text{ is ego.} \end{cases} \tag{14}$$

4.3 Leveraging proxy gradients Unlike the surprise, $h(\varphi)$, the *proxy surprise*, $\hat{h}$, is differentiable with respect to the pivot values $\mathbb{V}$. One can interpret $\nabla\hat{h}(\mathbb{V}^\varphi)$ then as suggesting how to modify the pivot values in order to make the demonstrations less surprising. A natural question then is how to adapt φ given this knowledge. Observe the following two propositions (with proof sketches provided in Sec 8):

Proposition 1 (Pivot values respect subsets). *Let ξ be a complete path that pivots at node i. If $\varphi \subsetneq \psi$ and $\xi \in \psi \setminus \varphi$, then $\mathbb{V}_i^\varphi < \mathbb{V}_i^\psi$.*

Proposition 2 ($\nabla\hat{h}$ determined by prefix tree path probabilities).

$$\frac{\partial \hat{h}}{\partial \mathbb{V}_k} = \sum_{\substack{(i,j)\in E \\ i \text{ is ego}}} \#_{(i,j)} \cdot \Big(\Pr(i \rightsquigarrow k \mid \mathbb{V}) - \Pr(j \rightsquigarrow k \mid \mathbb{V}) \Big) \tag{15}$$

Prop 1 suggests that to adjust the pivot value, $\mathbb{V}_i$, in a manner that decreases surprise, one can simply take a path, ξ, that pivots through i such that:

$$\xi \in \varphi \iff \frac{\partial \hat{h}}{\partial \mathbb{V}_i} > 0, \tag{16}$$

and negate its satisfaction under φ.

Prop 2 illustrates that (i) the gradients are simple to compute given black box access to the policy on the prefix tree and (ii) sampling from π_φ yields paths with a large effect on the gradient. To see point (ii), consider extending the prefix tree to contain the most likely paths after pivoting and apply Prop 2. The gradient

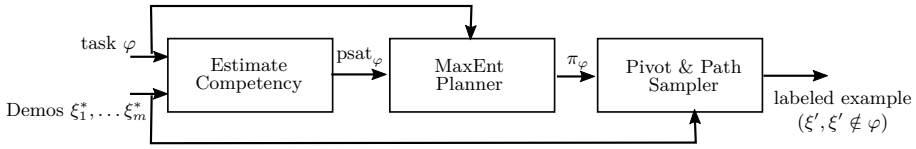


Fig. 6: Overview of surprise guided sampler.

Algorithm 1 Surprise Guided Sampler.

procedure SGS($\varphi, X, \mathcal{T}, M, \beta$)
  Compute π_φ given M and $\mathcal{T}$. ▶ MaxEnt Planning.
  Let D be the distribution over pivots s.t. $\Pr(\text{pivot } i) \propto \exp\left(-\frac{1}{\beta}\left|\frac{\partial \hat{h}}{\partial \mathbb{V}_i}\right|\right)$.
  Sample a pivot $i \sim D$ and a path $\xi \sim (\pi_\varphi, M)$ such that:
    a) ξ pivots at i.
    b) $\xi \in \varphi \iff \frac{\partial \hat{h}}{\partial \mathbb{V}_i} > 0$.
    c) $\exists \varphi' \in \Phi$ s.t. φ' is consistent with $X \cup \{(\xi, \xi \notin \varphi)\}$.
  return ξ ▶ Conjecture mis-labeled path.
end procedure

w.r.t. the pivot value corresponding to the newly introduced leaves is their path probability and thus their probability of being sampled after pivoting! Using these insights we propose surprise guided sampling (Alg 1) which samples a path to relabel based on (i) how likely it is under π_φ and (ii) the magnitude and sign of the gradient at the corresponding pivot. Combined with an identification algorithm, $\mathcal{I}$, repeated applications of Alg 1 yields an infinite (and stochastic) sequence of tasks resulting from incrementally conjecturing mis-labeled paths.

Remark 4. Alg 1 only *requires* a black box maximum entropy (MaxEnt) planner to enable assigning edge probabilities, $\Pr(i \rightsquigarrow j \mid \mathbb{V})$, and sampling suffixes given a pivot. If the satisfaction probability of an action is also known, i.e., $\Pr_{\xi'}(\xi \cdot \xi' \in \varphi \mid \xi, M, \pi_\varphi)$, then one can more efficiently sample suffixes using Baye's rule and the policy π_φ.

5 Demonstration Informed Specification Search (DISS)

In this section, we take the insights developed in the previous sections and propose the DISS algorithm - a variant of simulated annealing for quickly finding probable task specifications. At a high level, *Simulated Annealing* (SA) [12] is a probabilistic optimization method that seeks to minimize an energy function $U : Z \to \mathbb{R} \cup \{\infty\}$. To run SA, one requires three ingredients: (i) a *cooling schedule* which determines a monotonically decreasing sequence of temperatures, (ii) a *proposal* (neighbor) distribution $q(z' \mid z)$, and (iii) a *reset* schedule, which periodically sets the current state, z_t, to one of the lowest energy candidates seen so far.

A standard simulated annealing algorithm then operates as follows: (i) An initial $z_0 \in Z$ is selected (ii) T_t is selected based on the cooling schedule. (iii) A neighbor z' is sampled from $q(\cdot \mid z_t)$. (iv) z' is accepted ($z_{t+1} \leftarrow z'$) with probability:

$$\Pr(\text{accept} \mid z', z_t) = \begin{cases} 1 & \text{if } U(z') < U(z) \\ \min\left\{1, \exp\left(\frac{U(z)-U(z'))}{T_t}\right)\right\} & \text{otherwise} \end{cases}, \tag{17}$$

and rejected ($z_{t+1} \leftarrow z_t$) otherwise. (v) Finally, if a reset set is trigger, z_{t+1} is sampled from the minimum energy candidates, e.g., uniform on the argmin.

As previously stated, we propose a variant of simulated annealing adapted for our specification inference problem. We will start by assuming the posterior distribution on tasks takes the form:

$$\Pr(\varphi \mid \xi_1^*, \dots \xi_m^*, M) \propto e^{-U(\varphi)}, \tag{18}$$

where the *energy*, U, is given by:

$$U(\varphi) \stackrel{\text{def}}{=} \text{size}(\varphi) + \theta \cdot h(\varphi), \tag{19}$$

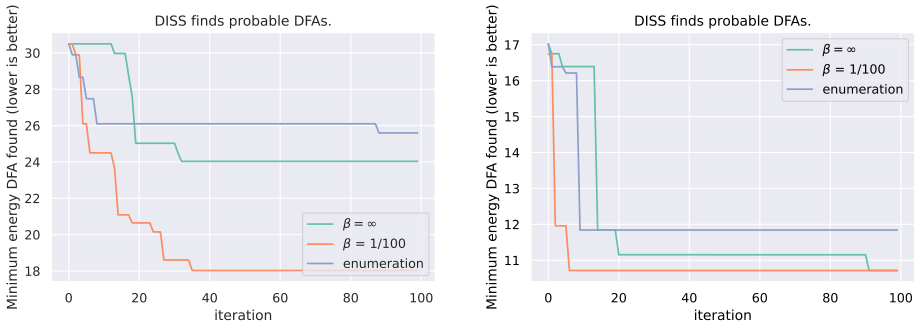
where $\theta \in \mathbb{R}$ determines the relative weight of the surprise. That is, we appeal to Occam's razor and assert that the task distribution is exponentially biased towards simpler tasks, where simplicity is measured by the description length of the task, size(φ), and the description length of $\xi_1^*, \dots \xi_m^*$ under (π_φ, M).

Remark 5. One might set $\theta > 1$ when the analysis is done for a high-level abstraction of the "true" MDP, and thus the low-level surprise, which includes the randomness of the low level controller, may be non-trivially larger per high-level time step.

Using the language of SA, we define DISS as follows: (i) $z \in Z$ is a tuple, (X, φ), of labeled examples and a task specification. (ii) $z_0 = (\emptyset, \bot)$. (iii) the proposal distribution, $q(X', \varphi' \mid X, \varphi)$ is defined to first sample a concept using an identification map, $\varphi' \sim \mathcal{I}(X)$, then run SGS on φ' to conjecture a labeled path ξ, yielding $X' = X \cup \{(\xi, \xi \notin \varphi')\}$. (iv) resets occur every $\kappa \in \mathbb{N}$ time steps. If a reset is triggered, X_{t+1} is sampled from $\text{softmin}_{i \le t} U(\varphi_i)$, and φ_{t+1} is sampled from $\mathcal{I}(X_{t+1})$.

6 Experiments

In this section, we illustrate the effectiveness of DISS by having it search for a ground truth specification, represented as a DFA, given the expert demonstrations, ξ_b, ξ_g, from our motivating example (shown in Fig 1). The (dotted) green path, ξ_g, goes directly ■. The (solid) black path, ξ_b, immediately slips into ■, visits ■, then proceeds towards ■. This path is incomplete, with a possible extension, σ_b, shown as a dotted line. The ground truth task is the right DFA in Fig 2.



(a) Minimum energy found by iteration in monolithic experiment.

(b) Minimum energy found by iteration in incremental experiment.

Fig. 7

We consider two specification inference problems by varying the concept class and the provided demonstrations. These variants respectively illustrate that (i) our method can be used to incrementally learn specifications from unlabeled incomplete demonstrations and (ii) the full specification can be learned given unlabeled complete demonstrations.

1. **Monolithic**: ξ_g and $\xi_b \cdot \sigma_b$ are provided as (unlabeled) *complete* demonstrations. The concept class is Φ_{reg} from Example 1.
2. **Incremental**: ξ_b is provided as an (unlabeled) *incomplete* demonstration. The concept class is Φ'_{reg} from Example 1.

The surprise weight, θ, is set to 1 for both variants. Finally, two additional inductive biases, which empirically proved necessary for optimizing the random pivot baseline, are applied: (i) we remove white tiles, □, from labeled examples (ii) we transform sequences of repeated colors into a single color thus biasing towards DFA that do not count. For example, □■■□■■ ↦ ■■■.

DISS parameters. Our implementation of DISS uses SGS temperature $\beta = 1/100$, resets every 10 iterations, and uses the following cooling schedule:

$$T_t = 100 \cdot (1 - t/100) + 1. \tag{20}$$

For concept identification, $\mathcal{I}$, we adapt an existing SAT-based DFA identification algorithm [13] to enumerate the first 20 consistent DFAs (ordered by size). A DFA is then sampled from $\text{softmin}_\varphi(\text{size}(\varphi))$. For maximum entropy planning, we use the Binary Decision Diagram based approached proposed in [15] with a planning horizon of 15 steps. Finally, because our experiments operate with one or two demonstrations, the competency of the maxent policy is taken to be closest achievable competency to $4/5$, e.g., for $\varphi = \emptyset$, the competency is 0, and for φ_1 the competency is $4/5$.

Baselines. As mentioned in the introduction, existing techniques for learning specifications from demonstrations use various syntactic concept classes, each

with their own inductive biases. To normalize the algorithms, we implemented two DFA adapted baselines that act as proxies for the enumerative and probabilistic hill climbing families of existing work:

1. **Prior-ordered Enumeration.** This baseline uses the same SAT-based DFA identification algorithm to find the first N DFAs, ordered by prior probability, i.e., size. As an alternative to DISS's competency assumption, we allow the enumerative baseline to restrict the search to task specifications that accept the provided demonstrations.[1]
2. **Random Pivot DISS.** This baseline uses DISS with SGS temperature, $\beta = \infty$. This ablation results in a (labeled example) mutation based search with access to the same class of mutations as DISS, but samples pivots uniformly at random, i.e., no gradient based bias. Note that this variant still samples suffixes conditioned on the sign of the gradient, and thus the mutations are still informed by the surprise.

6.1 Results and Analysis To simplify our analysis, we present time in iterations, i.e., number of sampled DFAs, rather than wall clock time. This is for two reasons. First, for each algorithm, the wall clock-time was dominated by synthesizing maximum entropy planners for each unique DFA discovered, but the choice of planner is ultimately an implementation detail. Second, because many DISS iterations correspond to the same DFAs (due to resets and rejections) the enumeration baseline explored significantly more *unique* DFAs than DISS (a similar effect occurs with the random pivot baseline, since it different pivots give more diverse example sets). Thus, using wall clock-time would skew the results below in DISS's favor.[2]

Fig 7a and Fig 7b show the minimum energy DFA for the monolithic and incremental experiments respectively. We see that for both experiments, DISS was able to significantly outperform the baselines (recall that energy is the negative log of the probability), with the incremental experiment requiring only a few ($<$ 6) iterations to find a probable DFA! Furthermore, in addition to finding the most probable DFAs much faster than the baselines, DISS also found *more* high probability DFAs.

Next we analyze the DFAs learned by DISS. The ground truth DFA (center) and the most likely DFAs found by DISS for each experiment (left and right) are shown in Fig 8. We observe that for both experiments, DISS is able to learn that if the agent visits ■, it needs to visit ■ before ■! Furthermore, the monolithic DFA is impressively able to learn that ■ leads to a sink state, a feat that requires quite a number of negative examples to illicit from our size-based DFA sampler. In fact, this discovery is responsible for the large drop in energy at 15 iterations in the monolithic experiment.

[1] For the incremental experiment, a counterexample loop is used add labeled examples that bias the DFAs towards implying φ_1.

[2] Nevertheless, for the monolothic experiment, the wall clock times for DISS, random pivoting, and enumeration were 542s, 764s, and 617s respectively. Similarly, for the incremental experiment the respective times were 353s, 464s, and 820s.

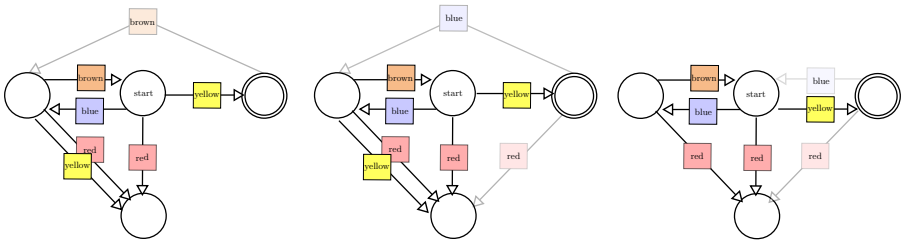


Fig. 8: (center) Ground truth DFA. Most probable DFA found by DISS in the monolithic experiment (left) and the incremental experiment (right).

Nevertheless, our learned DFAs differ from ground truth, particularly when it comes to the acceptance of strings *after* visiting ■. We note that a large reason for this is that our domain and planning horizon make the left most ■ effectively act as a sink state. That is, the resulting sequences are indistinguishable, with many even having the exact same energy. In Fig 8, we make such edges lighter, and note that the remainder of the DFAs show good agreement with the ground truth. Finally, while impressive, this points to a fundamental limitation of demonstrations. Namely, if two tasks have very correlated policies in a workspace, then without strong priors or side information, one is unable to distinguish the tasks. For example, if a task requires the agent to avoid ■, but no ■ are shown in the workspace, then one cannot hope to learn this aspect of the task.

7 Conclusion

This paper considered the problem of learning history dependent task specifications, e.g. automata and temporal logic, from expert demonstrations. We showed empirically demonstrated how to efficiently explore intractably large concept classes such as deterministic finite automata for find probable task specifications. The proposed family of algorithms, *Demonstration Informed Specification Search (DISS)*, requires only *black box* access to (i) a Maximum Entropy planner and (ii) an algorithm for identifying concepts, e.g., automata, from labeled examples. While we showed concrete examples for the efficacy of this approach, several future research directions remain. First and foremost, research into faster and model-free approximations of maximum entropy planners would enable a much larger range of applications and domains. Similarly, while large, the demonstrated concept class was over a small number of pre-defined atomic predicates. Future work thus includes generalizing to large symbolic alphabets and studying more expressive specification formalisms such as register automata, push-down automata, and (synchronous) products of automata.

8 Proof Sketches

Proof (Prop 1). Follows inductively from the monotonicity of $\mathbb{E}$, $\sum$, and ln. ■

Before proving Prop 2 we first prove the following lemma.

Lemma 1. *For any node, i, in the prefix tree,*

$$\frac{\partial}{\partial \mathbb{V}_k} \square_i(\mathbb{V}) = \Pr(i \rightsquigarrow k \mid \mathbb{V}),$$

where $\square$ denotes $\hat{V}$ for ego nodes and $\hat{Q}$ for env nodes.

Proof. For any edge (a, b), observe that if a is an environment node, then $\Pr(a \rightsquigarrow b \mid \mathbb{V})$ is a constant, denoted q_{ab}. Next, observe that because the nodes are arranged as a tree either: (1) k is not reachable from i or (2) only a single edge, call (i, j), can reach k from i. Thus,

$$\begin{aligned} \frac{\partial \hat{Q}_i}{\partial \mathbb{V}_k} &\stackrel{\text{def}}{=} \frac{\partial}{\partial \mathbb{V}_k} \sum_{\substack{(a,b)\in E \\ i=a}} q_{ib} \cdot \hat{V}_b(\mathbb{V}) \\ &= \Pr(i \rightsquigarrow j \mid \mathbb{V}) \cdot \begin{cases} 0 & \text{if } \Pr(i \rightsquigarrow k) = 0 \\ \frac{\partial}{\partial \mathbb{V}_k} \hat{V}_j(\mathbb{V}) & \text{otherwise,} \end{cases} \end{aligned} \tag{21}$$

Similarly, note that because the derivative of logsumexp is the softmax function, for any ego node i,

$$\begin{aligned} \frac{\partial \hat{V}_i}{\partial \mathbb{V}_k} &\stackrel{\text{def}}{=} \frac{\partial}{\partial \mathbb{V}_k} \log \sum_{\substack{(a,b)\in E \\ i=a}} \hat{Q}_b(\mathbb{V}) \\ &= \begin{cases} 0 & \text{if } \Pr(i \rightsquigarrow k) = 0 \\ e^{\hat{Q}_j(\mathbb{V}) - \hat{V}_i(\mathbb{V})} \cdot \frac{\partial}{\partial \mathbb{V}_k} \hat{Q}_j(\mathbb{V}) & \text{otherwise,} \end{cases} \end{aligned} \tag{22}$$

where again, j denotes the (potential) unique child of i that can reach k. Finally, observe that by definition $e^{\hat{Q}_j(\mathbb{V}) - \hat{V}_i(\mathbb{V})} = \Pr(i \rightsquigarrow j \mid \mathbb{V})$, using the maximum entropy policy induced by $\mathbb{V}$. Substituting into (22), we see that the lemma follows by induction. ■

Proof (Prop 2). Recall that the probability of traversing an environment edge is constant w.r.t $\mathbb{V}$. Thus, inspecting (14) we see that it suffices to prove that for any ego edge, (i, j),

$$\frac{\partial}{\partial \mathbb{V}_k} \ln \Pr(i \rightsquigarrow j \mid \mathbb{V}) = \Pr(i \rightsquigarrow k \mid \mathbb{V}) - \Pr(j \rightsquigarrow k \mid \mathbb{V}).$$

Recall that by definition, if i is ego, then $\ln \Pr(i \rightsquigarrow j \mid \mathbb{V}) = \hat{Q}_i(\mathbb{V}) - \hat{V}_i(\mathbb{V})$. Thus, the proposition follows directly from Lemma 1. ■

References

1. Carrillo, E.: Controller Synthesis and Formal Behavior Inference in Autonomous Systems. Ph.D. thesis, University of Maryland, College Park (2021)
2. Chou, G., Ozay, N., Berenson, D.: Explaining multi-stage tasks by learning temporal logic formulas from suboptimal demonstrations. In: Toussaint, M., Bicchi, A., Hermans, T. (eds.) Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020 (2020). https://doi.org/10.15607/RSS.2020.XVI.097, https://doi.org/10.15607/RSS.2020.XVI.097
3. Heule, M., Verwer, S.: Exact DFA identification using SAT solvers. In: Sempere, J.M., García, P. (eds.) Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6339, pp. 66–79. Springer (2010). https://doi.org/10.1007/978-3-642-15488-1_7, https://doi.org/10.1007/978-3-642-15488-1_7
4. De la Higuera, C.: Grammatical inference: learning automata and grammars. Cambridge University Press (2010)
5. Jaynes, E.T.: Information theory and statistical mechanics. Physical review **106**(4), 620 (1957)
6. Kalman, R.E.: When is a linear control system optimal (1964)
7. Kasenberg, D., Scheutz, M.: Interpretable apprenticeship learning with temporal logic specifications. arXiv preprint arXiv:1710.10532 (2017)
8. Luce, R.D.: Individual choice behavior. (1959)
9. Ng, A.Y., Russell, S.J., et al.: Algorithms for inverse reinforcement learning. In: ICML. pp. 663–670 (2000)
10. Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. IJCAI (2007)
11. Shah, A., Kamath, P., Shah, J.A., Li, S.: Bayesian inference of temporal task specifications from demonstrations. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. pp. 3808–3817 (2018)
12. Skiscim, C.C., Golden, B.L.: Optimization by simulated annealing: A preliminary computational study for the TSP. In: Roberts, S.D., Banks, J., Schmeiser, B.W. (eds.) Proceedings of the 15th conference on Winter simulation, WSC 1983, Arlington, VA, USA, December 12-14, 1983. pp. 523–535. ACM (1983), http://dl.acm.org/citation.cfm?id=801546
13. Ulyantsev, V., Zakirzyanov, I., Shalyto, A.: Bfs-based symmetry breaking predicates for DFA identification. In: Dediu, A., Formenti, E., Martín-Vide, C., Truthe, B. (eds.) Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings. Lecture Notes in Computer Science, vol. 8977, pp. 611–622. Springer (2015). https://doi.org/10.1007/978-3-319-15579-1_48, https://doi.org/10.1007/978-3-319-15579-1_48
14. Vazquez-Chanlatte, M., Jha, S., Tiwari, A., Ho, M.K., Seshia, S.A.: Learning task specifications from demonstrations. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. pp. 5372–5382 (2018)

15. Vazquez-Chanlatte, M., Seshia, S.A.: Maximum causal entropy specification inference from demonstrations. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12225, pp. 255–278. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_15, https://doi.org/10.1007/978-3-030-53291-8_15
16. Vazquez-Chanlatte, M., Shah, A.: Demonstration Informed Specification Search, https://github.com/mvcisback/DISS
17. Yoon, H., Sankaranarayanan, S.: Predictive runtime monitoring for mobile robots using logic-based bayesian intent inference. In: IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021. pp. 8565–8571. IEEE (2021). https://doi.org/10.1109/ICRA48506.2021.9561193, https://doi.org/10.1109/ICRA48506.2021.9561193
18. Ziebart, B.D., Bagnell, J.A., Dey, A.K.: Modeling interaction via the principle of maximum causal entropy (2010)
19. 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)

Algorithm 2 Demonstration Informed Specification Search.

procedure DISS($(\xi_1, \ldots, \xi_m), M, \theta, N, \kappa$)
  Compute $\mathcal{T}$ given $(\xi_1, \ldots, \xi_m)$. ▶ Create prefix tree.
  $\Phi \leftarrow \emptyset$.
  for t in $1, \ldots, N$ **do**
    if $t \equiv 0 \pmod{\kappa}$ **then** ▶ Reset on multiples of κ.
      $X \sim \arg\max_{\psi \in \Phi} U(\psi)$
      $dX \leftarrow \emptyset$ ▶ Defined as $\emptyset$ if $\Phi = \emptyset$.
    end if
    $X' \leftarrow \text{update}(X', dX)$ ▶ Newer label wins under conflict.
    $\varphi' \sim \mathcal{I}(X')$. ▶ Sample candidate task.
    $\Phi \leftarrow \Phi \cup \{\varphi'\}$. ▶ Update visited specs.
    $T \leftarrow \text{cooling\_schedule}(t)$ ▶ User defined.
    $dU \leftarrow U(\varphi') - U(\varphi)$
    $\alpha \sim \text{Uniform}(0, 1)$
    if $dU < 0$ or $\exp(-^{dU}/_{T}) \leq \alpha$ **then**
      $(\varphi, X) \leftarrow (\varphi', X')$
      $dX \leftarrow \{\text{SGS}(\varphi, T, M)\}$ ▶ Conjecture labeled example.
    else
      $dX \leftarrow \emptyset$ ▶ Reject proposal.
    end if
  end for
  return Φ
end procedure