Learning Formal Specifications from Membership and Preference Queries

Ameesh Shah [1] **Marcell Vazquez-Chanlatte** [1] **Sebastian Junges** [2] **Sanjit A. Seshia** [1]

Abstract

Active learning is a well-studied approach to learning formal specifications, such as automata. In this work, we extend active specification learning by proposing a novel framework that strategically requests a combination of membership labels *and pair-wise preferences*, a popular alternative to membership labels. The combination of pair-wise preferences and membership labels allows for a more flexible approach to active specification learning, which previously relied on membership labels only. We instantiate our framework in two different domains, demonstrating the generality of our approach. Our results suggest that learning from both modalities allows us to robustly and conveniently identify specifications via membership and preferences.

1. Introduction

An emerging body of work advocates for the use of *formal specifications* to model objectives of autonomous agents. Formal specifications provide a number of benefits relative to Markovian rewards: they can specify non-Markovian historical dependencies, they are composable, and they can be easily transferred and understood across environments. Such specifications are popular in planning, verification, and robotics (Webster et al., 2020; Yifru & Baheri, 2023).

A popular approach to learning formal specifications is *active* learning, querying a non-human membership oracle to label a generated trajectory as either positive (belongs in the set of desired behaviors) or negative (Bastani et al., 2018; Bongard & Lipson, 2005; Angluin, 1987). We aim to extend these approaches to cases where we learn from non-expert human teachers.While asking only membership queries suffices for completeness in active specification learning, we expect existing approaches to be challenging for human oracles, who are relatively bad at answering membership queries (Palan et al., 2019; Burton et al., 2021; Phelps et al., 2015).

To address this challenge, we identify *preference querying* as a promising alternative to membership querying, where the human is asked to rank two trajectories. Preferences are relatively inexpensive to obtain, are generally preferred by human teachers, and are known to be less susceptible to mislabeling than membership query responses (Palan et al., 2019; Burton et al., 2021; Phelps et al., 2015). Moreso, the combination of the two signals is promising: *Preference queries are comparatively more accurate but less informative than membership queries* (Palan et al., 2019).

In this work, we contribute a general framework that allows for active learning from a combination of preferences and labeled examples. The framework works as follows: First, from the known facts, it generates candidate specifications consistent with previously observed membership and preference constraints. The next step is to rule out hypotheses from this candidate set that are consistent with the known information, but incorrect. This step is realized by actively asking either a preference query or a membership query chosen to gain information that will rule out incorrect candidate hypotheses. From this point, we iteratively generate new candidate hypotheses that are consistent with the newly updated facts, and continue asking queries until the correct specification is found, if one exists. To robustify the algorithm against a (limited) number of wrong answers, the algorithm can identify and ignore inconsistent sets of answers.

Our experiments show that our automated query selection process can avoid a substantial number of membership queries by asking additional preference queries. This trade-off between queries can be easily configured by setting the relative cost of answering each type of query to the teacher.

Contributions We present the formalism of membership-preserving preferences that allows for specification learning from oracles that can answer comparison and membership queries. To show the feasibility and efficacy of our formalism, we propose a *concept class agnostic* algorithm for querying oracles and run empirical evaluations on two different domains. Finally, we provide a novel SAT-based encoding for identifying DFAs from labeled examples *and* pair-wise preferences.

[1]UC Berkeley [2]Radboud University. Correspondence to: Ameesh Shah <ameesh@berkeley.edu>.

The Many Facets of Preference Learning Workshop at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023. Copyright 2023 by the author(s).

2. Learning with Membership Respecting Preferences

We begin by developing the machinery to specify which behaviors within a set are desirable, both in a relative and satisficing sense. For a given *universe*, $\mathcal{U}$, containing *atoms* $x \in \mathcal{U}$, a formal specification, or *concept* $\varphi \subseteq \mathcal{U}$, contains a set of atoms. We denote by $\varphi(x)$ the indicator, $[x \in \varphi]$ and refer to a collection of concepts, Φ, as a *concept class*[1]. W.l.o.g., we assume that the universe coincides with the union of the concepts in a concept class, $\mathcal{U} = \bigcup \Phi$.

Example 1. Concept classes and their universes can be finite or infinite. For example, when representing formal task specifications, one takes as the (infinite) universe, Σ^*, i.e., all words from a finite alphabet Σ. Similarly, languages represented by classes of automata, e.g., DFAs, are (infinite) concept classes.

Next, we formalize the notion of pair-wise preferences on atoms via *preorders*, $\preceq$, on universes, i.e., transitive and reflexive relations on $\mathcal{U}$. Namely, we call a preorder on a universe a *preference order* and interpret $x \preceq y$ as *"y may be preferred over x"*. We write $x \prec y$ if $x \preceq y$ and not $y \preceq x$, interpreted as *"y is preferred over x"*. Two atoms have equal preference, $x \equiv y$, if $(x \preceq y) \wedge (y \preceq x)$. Finally, two atoms are *incomparable*, $x \mid\mid y$, if $\neg(x \preceq y \vee y \preceq x)$.

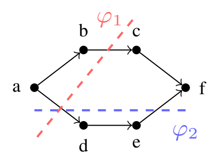

Figure 1.

Example 2. An example preference order over the universe $\{a, b, c, d, e, f\}$, is shown in Fig 1. It is represented using a directed acyclic graph, $H = (V, E)$ called a *Hasse diagram*. The nodes of H represent equivalence classes, i.e., for all $v \in V$, $x, y \in v \implies x \equiv y$, and the edges of H represent strict preferences, i.e., $(x, y) \in E \implies x \prec y$. The full preference order is the transitive reduction of H.

Example 3. Costs and rewards offer a common way to define preference orders. For example, let $x \in \mathcal{U}$ denote the set of paths through a maze and assign a cost, $c(x) \in \mathbb{R}$, to each path based on its length. A natural preference order is then given by comparing costs: $c(x) \leq c(y)$ implies $y \preceq x$. This order is total (no atoms are incomparable) and illustrates that two distinct atoms can have equal preference, i.e., $c(x) = c(y)$ implies $x \equiv y$, but not $x = y$.

2.1. Learning with Preferences

We now turn our attention to identifying an unknown concept, φ^*. Namely, we shall assume access to a *membership oracle*, $\mathcal{M} : \mathcal{U} \to \{\in, \notin\}$, to evaluate if $x \in \varphi^*$, as well as a comparison oracle, $\mathcal{C} : \mathcal{U}^2 \to \{\prec, \mid\mid, \succ, \equiv\}$, to provide preferences between atoms, e.g., $[C(x, y) = \prec]$ iff $[x \prec y]$. Invocations of these oracles are *queries*.

[1]For simplicity, we conflate a concept with its representation.

Problem Statement: Let φ be an *unknown* specification in concept class Φ_{init}. Given membership and comparison oracles $\mathcal{M}$ and $\mathcal{C}$, infer φ.

Remark 2.1. For finite concept classes it suffices to consider only the membership oracle $\mathcal{M}$, ignoring query and time complexity. In particular, one may pair-wise consider all concepts and pose a membership query from the symmetric difference of the concepts to uniquely identify the concept. However, for many domains (Burton et al., 2021), obtaining accurate labels to realize the membership oracles is expensive. The key question in this work is *how to exploit the availability of the comparison oracle $\mathcal{C}$.*

To leverage the comparison oracle, we relate preferences to membership in φ^*. We therefore focus on preference orders that respect membership: atoms outside of the concept, $x \notin \varphi^*$, cannot be preferred to atoms in the concept, $y \in \varphi^*$.

Definition 2.2. A preference order is a *membership-respecting preference* (MemReP) w.r.t. φ if

$$x \preceq y \implies \varphi(x) \leq \varphi(y). \tag{1}$$

Example 4. All preference orders are membership respecting w.r.t. concepts $\top \stackrel{\text{def}}{=} \mathcal{U}$ and $\bot \stackrel{\text{def}}{=} \emptyset$. Similarly, any oracle that yields $C(x, y) = \mid\mid$ for all $x, y \in \mathcal{U}$ is vacuously membership-respecting.

Example 5. Cost based preferences, like in our path example, together with thresholded cost concepts, i.e., $\varphi_\delta \stackrel{\text{def}}{=} \{x \in \mathcal{U} : c(x) \leq \delta\}$, for some cost map $c : \mathcal{U} \to \mathbb{R}$ and threshold $\delta \in \mathbb{R}$, are membership respecting.

Example 6. We continue with Ex. 2. This order is membership-preserving for $\varphi_1 = \{a, b\}$, but not for $\varphi_2 = \{d, e\}$. Graphically, φ_2 is not membership-preserving as there is an edge from $\bar{\varphi}_2$ to φ_2.

Example 6 illustrates that the MemReP assumption is sometimes strong enough to distinguish concepts without any membership queries. Namely, if there exists a pair of atoms, $x \in \varphi_1 \setminus \varphi_2$ and $y \in \varphi_2 \setminus \varphi_1$, such that $\mathcal{C}(x, y) \neq \mid\mid$, then $\mathcal{C}(x, y)$ is guaranteed to distinguish φ_1 and φ_2 under the MemReP assumption.

Definition 2.3. Let $X = (X_\mathcal{M}, X_\mathcal{C})$ be a tuple of sets of membership and comparison labeled examples respectively, e.g.,

$$X_\mathcal{M} = \{(x_1, \in), \ldots, (x_j, \notin)\}$$
$$X_\mathcal{C} = \{(x_k, y_k, \mid\mid), \ldots, (x_n, y_n, \prec)\}.$$

A concept φ is *consistent* with X if (i) φ agrees with all membership assignments in $X_\mathcal{M}$, e.g., $(x, \in) \in X_\mathcal{M} \implies x \in \varphi$; and (ii) All preferences in $X_\mathcal{C}$ are membership respecting under φ, e.g., $(x, y, \prec\ \varphi(x) \leq \varphi(y)$. Given a concept class Φ, we denote by Φ^X the set of all concepts in Φ consistent with X.

2.2. Abstract Algorithm

We now return to the problem of identifying an unknown concept given access to a membership oracle and a membership respecting comparison oracle. In Alg. 1, we outline the general process for learning concepts from *finite* concept classes given such oracles. The learner begins with a concept class, asks a membership or comparison query, and then removes concepts that are inconsistent with the query result until a unique concept remains.

Algorithm 1 Generic algorithm for concept identification.

1: Assign Φ as the initial concept class Φ_{init}.
2: Initialize the set of labeled examples X as $(\emptyset, \emptyset)$.
3: **while** $|\Phi| > 1$ **do**
4: Ask either a membership or a comparison query given X and Φ^X.
5: Add the result to X.
output Φ^X $\{\Phi^X = \emptyset$ or $\Phi^X = \{\phi^*\}.\}$

Proposition 2.4. *Suppose for every iteration, the probability of asking a distinguishing membership query, i.e., asking $\mathcal{M}(x)$ for x in the symmetric difference of two concepts in Φ^X, is bounded from below. Then, Alg 1 almost surely terminates.*

Remark 2.5. To treat infinite concept classes, we appeal to Occam's Razor, and seek to find the "simplest" concept that is consistent with the data. Formally, we assume that the finite concept class is the (countably infinite) union of finite concept classes, $\Phi = \bigcup_{i=1}^{\infty} \Phi_i$, where i, is taken as a complexity measure, e.g, number of states of an automaton. We extend the above process by seeking to find the smallest i such that the above process returns a singleton. Finally, as is standard in this setting, we will additionally assume access to an equivalence oracle to provide completeness guarantees. In practice, such equivalence queries are often impractical, and are approximated via conformance testing and sampling.

In order to realize this algorithm in practice, we require three ingredients. First, we must develop methods that can synthesize a consistent concept over both membership query and preference query results for a specific concept class. This operation enables symbolically interacting with Φ^X in Alg. 1. We provide synthesis methods for the concept classes used in our experiments in the appendix. Second is an intelligent means to select a query for $\mathcal{M}$ or $\mathcal{C}$, which we expand on in the sequel. Last is a means to non-trivial possibility of labeling mistakes, which we provide in the appendix.

3. Asking the right queries

In the sequel, we discuss a (concept class agnostic) strategy to select queries for an efficient version of Alg. 1.

3.1. Cost model

Before we optimize our algorithm, we must set the measure that we aim to optimize. In any active learning algorithm, the selection of the queries is central to its performance. In particular, we seek to balance minimizing the number of queries asked and computational costs. To this end, we model the costs of preference and membership queries to be time invariant and constant, based on a weighted sum:

$$\text{cost(queries)} \triangleq a \cdot \#\text{mem} + b \cdot \#\text{pref}, \tag{2}$$

where #mem and #pref refer to the number of membership and preference queries and $a, b \in \mathbb{R}_\infty$. For example, $a = b$ treats membership and comparisons interchangeably and $a = \infty, b = 1$ lexicographically prefers comparisons over membership queries[2].

3.2. Contextual Bandit Formulation

Because the results of the queries are a priori unknown, naïvely optimizing a given cost model is often infeasible. Furthermore, as the next example illustrates, adversarial teachers can induce arbitrary regret.

Example 7. Recall the lexicographic example, $a = \infty, b = 1$. If all preferences yield incomparable, then one would regret making any comparison queries. On the other hand, if one ignores comparisons, but the underlying preference lattice is total as in Ex. 5, then one may ask many more membership queries than is required. In particular, if the (total) preference order were known, then the learner could binary search (using $\mathcal{M}$) for the unique point where membership changes.

Further, we highlight that, even if a model for query responses is known, optimally planning a sequence of queries is often computationally intractable. There are $M = \binom{|\mathcal{U}|}{2} + \mathcal{U}$ queries to ask in a single step, and thus roughly 3^{M^t} possible combinations when asking up to t queries. Thus, even assuming $\mathcal{U}$ is finite, but non-trivially small, e.g., strings of length at most 10, the search space quickly becomes intractable.

Thus, we propose a two stage formulation to optimize the query selection based on adversarial contextual multi-armed bandits (CMABs) (Auer et al., 2002), specifically, multi-armed bandits with expert advice. In this formulation, the classic multi-armed bandit framework is extended by introducing *experts* that consider the arms and context in the problem, and provide recommendations in the form of probability distributions over the arms that our algorithm can then make use of in its policy. In order to instantiate our CMAB algorithm with experts, we make the following choices: (i) We use a heuristic to select candidate compari-

[2]Our algorithm can handle arbitrary cost models over #mem and #pref and is not dependent on the specific structure of equation 2.

Algorithm 2 CMAB round for selecting a query.

Select comparison and membership queries via Alg. 3.
Estimate the loss (3) for different query outcomes.
Provide average and worst-case expert advice distributions on the arms (queries), E_1, E_2, see Sec. 3.3.
Sample which expert, $E \sim \{E_1, E_2\}$, to listen to based on the historical performance of the expert.
Sample the arm (query) based on E's arm distribution.
Compute the actual loss and update expert distributions.

Algorithm 3 Query selection heuristic.

Select a set of up to α unrefuted concepts, $\Psi \subseteq \Phi$.
Let X be a set of atoms such that (i) $|X| \in [\alpha, \beta]$ and (ii) X distinguishes concepts in Ψ, i.e.

$$\forall \varphi_1, \varphi_2 \in \Psi \,.\, \exists x \in X \,.\, x \in \varphi_1 \Delta \varphi_2.$$

Find $x \in X$ that minimizes the (worst-case) number of concepts in Ψ that are consistent after $\mathcal{M}(x)$.
Find $y, z \in X$ that minimize the (worst-case) number of concepts in Ψ that are consistent after $\mathcal{C}(y, z)$.
Return candidate queries $\langle \mathcal{M}(x), \mathcal{C}(y, z) \rangle$.

son and membership queries, the *arms* in our formulation. The heuristic is a (Monte Carlo) estimate of the concept class size's reduction for each query's possible outcome. (ii) We define a (bounded) proxy cost per arm weighing the query cost against the (estimated) reduction in the concept class:

$$\text{loss}_c \overset{\text{def}}{=} \frac{c}{\max(a, b)} \cdot \frac{|\Phi'|}{|\Phi|} \in [0, 1], \tag{3}$$

where c is the cost of selected arm's query type (thus either a or b). Finally, (iii) we encode two heuristic query strategies as *experts* assigning probabilities to each arm, described in Section 3.3. The resulting CMAB game proceeds in rounds described in Alg. 2, each round corresponds to an iteration in Alg. 1.

To avoid considering all atoms in $\mathcal{U}$, we propose the concept class agnostic heuristic in Alg. 3 to select candidate membership and comparison queries. For our implementation, we use $\alpha = 2$, and take at most two atoms per concept, i.e., $\beta = 2\alpha$.

3.3. Worst and average case advice

We propose to give advice towards either of the two queries based on combining two perspectives. In particular, we propose to use the following two experts with the following advice arm distributions: **(1) Pessimistic**: Weighs each arm by its worst-case loss. Ignores the incomparable answer for preference queries. **(2) Historical**: Weighs each arm by its expected loss, computed by averaging the previous losses incurred by pulling that particular arm.

To construct the advice distribution, we take the softmax of the weights of the comparison and membership queries. The expert selection distribution is then updated using the standard exp4 contextual bandit algorithm (Auer et al., 2002) which guarantees that we will switch between our pessimistic and historical arm strategy in a manner such that, with respect to loss (3), the algorithm would not have done much better sticking to advice from a single expert.

Let us share the following intuitions for our choice of experts. The pessimistic expert helps to ensure some notion of progress. However, as an incomparable answer for the comparisons never yields any progress, we exclude this case from the expert. On the other hand, we have seen in Ex. 7 that the utility of a comparison query depends heavily on the oracle's underlying preference order. To capture this observation, the historical expert learns whenever almost all comparisons yield incomparable and will promote using membership in these cases. Likewise, this expert will discover when there is a total preference order and all atoms are comparable, and perhaps even equivalent. Finally, observe the following proposition about the proposed algorithm:

Proposition 3.1. *Assume the unknown concept, φ^*, is in Φ. Running Alg 1 using Algs 3,2 for queries (almost surely) identifies φ^* after finite queries.*

A proof sketch for Prop. 3.1 is provided in the appendix.

4. Experiments

In this section, we instantiate Alg. 1 on two families of concept classes, demonstrating the flexibility and efficacy of the proposed formalism and heuristics.

4.1. Deterministic Finite Automata

To begin, we apply the algorithm and operations outlined in Sec. 3 to the space of DFA identification, relying on the performance of modern Boolean satisfiability (SAT) solvers.

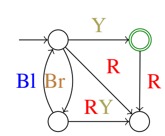

Figure 2.

A *Deterministic Finite Automaton*, DFA, is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of a finite set Q of *states*, a finite *alphabet* Σ, a *transition function*, $\delta \colon Q \times \Sigma \to Q$, an *initial state* q_0, and the *accepting states* $F \subseteq Q$. The function $\delta^* \colon \Sigma^* \to Q$ denotes the lifting of δ to sequences of symbols (strings), via repeated application. Finally, the language of DFA $\mathcal{D} = \{w \in \Sigma^* \mid \delta^*(w) \in F\}$ is set of strings that reach an accepting state, $q \in F$.

Learning Task Specifications for Robots We study the problem of conveying a task to an agent (robot) moving about an environment. For example, consider the task specification that says "avoid red (lava) tiles and do reach a yellow (recharge) tile (RY)" and "between visiting blue (water) tiles and yellow tiles, the agent must be visit a brown

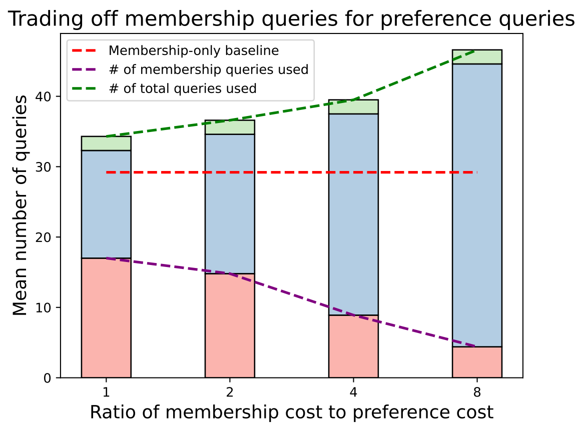

Figure 3. Trade-off between query types for DFAs. The bars show the contribution of membership (red), preference (blue), and equivalence (green) queries.

(dryer) tile. (BBY)" from (Vazquez-Chanlatte et al., 2018). This task specification is described by the DFA shown in Fig 2. We assume that the robot is pre-programmed to universally assume that a task will contain the RY constraint and seeks to interactively learn the domain-specific BBY constraint (in the form of a DFA) from a user.

First, observe that because DFAs are closed under conjunction, a new DFA can be easily derived that will not violate the a priori specified RY task. Further, depending on the system dynamics, the user may prefer some traces to others, and is able to provide these pair-wise comparisons. Thus, we pose this task learning process as an active learning problem, where the user can provide (i) *pair wise preferences* and (ii) *membership query responses (labels)* that specify whether an example is good or bad. We want to leverage that query mechanism to potentially accelerate or robustify the learning of the DFA.

The oracle uses a randomly generated membership respecting preference order, where approximately $1/10$ of the comparisons yield incomparable. To refute equivalence queries, we sample a random string from the symmetric difference of the true DFA and the current hypothesis. Finally, we fix the comparison cost to 1 and vary the membership cost to study the trade off in queries the algorithm makes.

The results of the experiment are shown in Fig 3. We observe that, as expected, as the membership cost increases (i) the total number of queries increases (ii) the total number of membership queries decreases. Specifically, *removing 1 membership query adds an average of 2.03 preference queries* each time the cost doubles. Additionally, the use of preference queries allows the use of membership queries to be well below the baseline rate of setting the costs such that only membership queries are used; that is, setting the cost of a preference to ∞.

Finally, we compare with the discriminate tree variant of L* (Kearns & Vazirani, 1994; Angluin, 1987), a classic algorithm for learning DFAs from membership and equivalence queries. We use the AALpy library (Muškardin et al., 2022) implementation of L*, and find that the implementation requires 11 membership queries and 2 equivalence queries (the same number as Fig 4) that cannot be answered by the RY task prior knowledge. While our algorithm requires more overall queries, we observe that the number of membership queries can be fewer than 11 as at membership cost 4, all without concept class specific tailoring. Furthermore, we remark that these two algorithms could be used in concert by adapting the membership query selection to use L* provided queries.

Additional Experiments In order to further study the performance of our algorithm in learning DFAs, we performed the following studies: First, we applied our algorithm to learning the Tomita Language DFAs, a standard benchmark in DFA learning, to evaluate performance across a diversity of DFAs. Next, we evaluated our algorithm on classes of DFAs that vary in size to evaluate how our algorithm scales as the target DFA increases in complexity. Finally, we implemented and evaluated a method in our algorithm that can identify and tolerate incorrect responses from our oracle. The full details and results of these experiments are available in the appendix.

4.2. Monotone Predicate Families

Next, we study monotone predicate families. A *monotone predicate family* is a concept class with an (arbitrary but fixed) partial order $\sqsubset$ defined over the concepts such that $\varphi \sqsubset \varphi'$ implies $\varphi \subseteq \varphi'$. Increasing a concept thus monotonically increases the set of atoms included by the concept. We motivate studying monotone predicates using a series of motivating examples.

Example 8. We consider the on-boarding process of a hypothetical car that queries the user to learn what (safe) distances to other objects they deem comfortable. In the scope of this paper: (a) The resulting behavior should *never* violate any pre-defined safety constraints. (b) The on-boarding experience should be brief, i.e., the system should try to minimize the number of queries. (c) Communication should be unambiguous and concrete to cover edge cases.

The be above example can be cast as a 2-dimensional monotone predicate family, where one dimension corresponds to maximum time, $\tau \in [0, T]$, the user is willing to wait to reach the destination and the other dimension corresponds to the minimum distance, $d \in [0, D]$, to another car the user is comfortable with. The corresponding partial order has $(\tau, d) \sqsubset (\tau', d')$ if $\tau < \tau'$ and $d > d'$.

We seek to understand the trade-off between membership and comparisons incurred by our algorithm given different

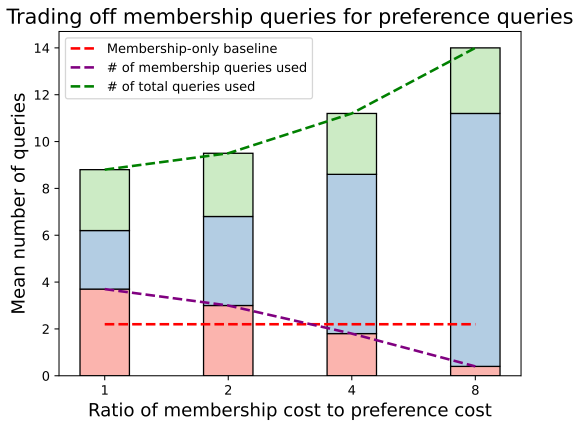

Figure 4. Trade-off between query types for monotone predicate functions. The bars show the contribution of membership (red), preference (blue), and equivalence (green) queries.

membership and comparison costs. We instantiated our algorithm with a size-indexed concept class, where each Φ_i corresponds to a uniform 2d grid of parameters with i points per axis. Equivalence queries provide a labeled bi-partition separating the concept. Like with our DFA experiment, the preference order was randomly generated such that approximately $^1/_{10}$ on atoms are incomparable and approximately $^1/_3$ of atoms whose preference is not forced by the MemReP condition are strictly ordered. Furthermore, as a baseline, we compare against an learner that only uses equivalence and membership queries by setting the comparison query cost sufficiently high.

Fig. 4 shows the averaged results on 100 randomly generated concepts with the comparison cost fixed to 1 and the membership cost changing on the horizontal-axis. The vertical axis corresponds to the average of number of queries across all preference orders. First, the average number of equivalence queries was the same for each instance on the same concept, including the shown membership only base-line. Furthermore, as desired, increasing the cost of the membership queries results in the average number of membership queries decreasing, at the expense of additional comparison queries: as the relative cost doubles, *removing 1 membership query adds on average 2.4 membership queries*. This is expected given that comparisons provide less information about the concept's label than a membership query.

We also observe that initially, introducing preferences occasionally *increases* the number of membership queries. This effect is due to: (i) the greedy nature of our algorithm, meaning that "good" membership queries are ignored as they correlate with "good" preference queries, and (ii) our hyperparameters - namely the temperature of the softmax in the expert advice - was tuned with the assumption that membership would cost significantly more than preferences.

5. Related Work

This work is related to (i) Learning reward from preferences and (ii) Grammatical inference and concept learning.

Active learning of rewards using preferences. Inverse reinforcement learning (IRL) (Ng & Russell, 2000; Abbeel & Ng, 2004) often relies on high-quality demonstrations to learn a reward function. More recently, works have proposed approaching IRL from an active learning perspective, asking a teacher for information-rich feedback on learner-generated examples, such as corrected examples or labels on sections of generated examples (Hadfield-Menell et al., 2016; Brown et al., 2018). Preference-based reward learning has emerged from these active approaches as a popular method due to the accuracy and relative inexpensiveness of preference queries (Holladay et al., 2016; Wilson et al., 2012). To overcome the limited information content gained from relative comparisons, various techniques have been devised to actively select preference queries that maximize the amount of information gained (Sadigh et al., 2017; Biyik & Sadigh, 2018; Xu et al., 2017; Basu et al., 2019; Xu et al., 2020), typically by removing maximal volume from the hypothesis space.

Grammatical inference and concept learning. Grammatical inference (De la Higuera, 2010) refers to the rich literature on learning a formal grammar (often an automaton) from data. Examples include learning the smallest automata consistent with a set of positive and negative strings (De la Higuera, 2010) or learning an automaton using membership and equivalence queries (Angluin, 1987). Within this literature, the key contributions of our work are to take into account preferences via the concept-agnostic framework. Finally, our algorithm can be seen as an extension of version space learning (Sverdlik & Reynolds, 1992), where we use preference-based learning to build on existing methods that leverage either explicit data structures or SAT-based DFA-identification (Ulyantsev et al., 2015; Heule & Verwer, 2010) to realize candidate elimination.

6. Conclusion and Future Work

In this paper, we present a generic framework for learning task specifications (concepts) from actively acquired from (noisy) preferences and labeled examples. Despite being concept class agnostic, we demonstrated the efficacy of our approach on two very different concept classes.

Nevertheless, interesting future work includes considering principled approaches to deriving domain-specific optimizations of the heuristics used in our framework. Furthermore, we hope to consider more expressive concept class families such as symbolic automata and context-free grammars: With symbolic automata we hope to support a mix of thresholded rewards and DFAs.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *ICML*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

Angluin, D. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

Bastani, O., Sharma, R., Aiken, A., and Liang, P. Active learning of points-to specifications. In Foster, J. S. and Grossman, D. (eds.), *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018*, pp. 678–692. ACM, 2018. doi: 10.1145/3192366.3192383. URL https://doi.org/10.1145/3192366.3192383.

Basu, C., Bıyık, E., He, Z., Singhal, M., and Sadigh, D. Active learning of reward dynamics from hierarchical queries. In *IROS*, pp. 120–127, 2019. doi: 10.1109/IROS40897.2019.8968522.

Biyik, E. and Sadigh, D. Batch active preference-based learning of reward functions, 2018.

Bongard, J. C. and Lipson, H. Active coevolutionary learning of deterministic finite automata. *J. Mach. Learn. Res.*, 6:1651–1678, 2005. URL http://jmlr.org/papers/v6/bongard05a.html.

Brown, D. S., Cui, Y., and Niekum, S. Risk-aware active inverse reinforcement learning. In *CoRL*, volume 87 of *Proceedings of Machine Learning Research*, pp. 362–372. PMLR, 2018.

Burton, N., Burton, M., Fisher, C., Peña, P. G., Rhodes, G., and Ewing, L. Beyond likert ratings: Improving the robustness of developmental research measurement using best–worst scaling. *Behavior Research Methods*, Apr 2021. ISSN 1554-3528. doi: 10.3758/s13428-021-01566-w. URL https://doi.org/10.3758/s13428-021-01566-w.

De la Higuera, C. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., and Dragan, A. D. Cooperative inverse reinforcement learning. In *NeurIPS*, pp. 3909–3917, 2016.

Heule, M. and Verwer, S. Exact DFA identification using SAT solvers. In *ICGI*, 2010.

Holladay, R., Javdani, S., Dragan, A., and Srinivasa, S. Active comparison based learning incorporating user uncertainty and noise. In *Proceedings of RSS Workshop on Model Learning for Human-Robot Communication*, June 2016.

Kearns, M. J. and Vazirani, U. V. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

Muškardin, E., Aichernig, B., Pill, I., Pferscher, A., and Tappler, M. Aalpy: an active automata learning library. *Innovations in Systems and Software Engineering*, 18: 1–10, 03 2022. doi: 10.1007/s11334-022-00449-3.

Ng, A. Y. and Russell, S. J. Algorithms for inverse reinforcement learning. In *ICML*, pp. 663–670. Morgan Kaufmann, 2000.

Palan, M., Shevchuk, G., Landolfi, N. C., and Sadigh, D. Learning reward functions by integrating human demonstrations and preferences. In *Robotics: Science and Systems*, 2019.

Phelps, A. S., Naeger, D. M., Courtier, J. L., Lambert, J. W., Marcovici, P. A., Villanueva-Meyer, J. E., and MacKenzie, J. D. Pairwise comparison versus likert scale for biomedical image assessment. *AJR. American journal of roentgenology*, 204(1):8–14, Jan 2015. ISSN 1546-3141. doi: 10.2214/AJR.14.13022.

Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. Active preference-based learning of reward functions. In *RSS*, 2017.

Sverdlik, W. and Reynolds, R. G. Dynamic version spaces in machine learning. In *ICTAI*, pp. 308–315. IEEE CS, 1992.

Tomita, M. Learning of construction of finite automata from examples using hill-climbing : Rr: Regular set recognizer. 1982.

Ulyantsev, V., Zakirzyanov, I., and Shalyto, A. Bfs-based symmetry breaking predicates for DFA identification. In *LATA*, volume 8977 of *LNCS*, pp. 611–622. Springer, 2015.

Vazquez-Chanlatte, M., Jha, S., Tiwari, A., Ho, M. K., and Seshia, S. A. Learning task specifications from demonstrations, 2018.

Webster, M., Western, D. G., Araiza-Illan, D., Dixon, C., Eder, K., Fisher, M., and Pipe, A. G. A corroborative approach to verification and validation of human-robot teams. *Int. J. Robotics Res.*, 39(1), 2020. doi: 10.1177/0278364919883338. URL https://doi.org/10.1177/0278364919883338.

Wilson, A., Fern, A., and Tadepalli, P. A bayesian approach for policy learning from trajectory preference queries. In *NIPS*, pp. 1142–1150, 2012.

Xu, Y., Zhang, H., Singh, A., Miller, K., and Dubrawski, A. Noise-tolerant interactive learning from pairwise comparisons, 2017.

Xu, Y., Chen, X., Singh, A., and Dubrawski, A. Thresholding bandit problem with both duels and pulls, 2020.

Yifru, L. and Baheri, A. Joint learning of policy with unknown temporal constraints for safe reinforcement learning. *CoRR*, abs/2305.00576, 2023. doi: 10.48550/arXiv.2305.00576. URL https://doi.org/10.48550/arXiv.2305.00576.

Appendix

A. MemRePs for DFAs

In order to realize our algorithm for DFA learning, we need to support synthesizing a DFA consistent with *labeled examples* V_+, V_- and an observed set of preferences, $V_\prec$, i.e. the results of previously observed membership and comparison queries respectively.

We extend the SAT encoding presented in (Ulyantsev et al., 2015; Heule & Verwer, 2010) for the passive identification of DFAs from positive and negative examples to support membership respecting preferences. The encodings operate by using the provided positive and negative examples to form a prefix tree. The nodes indexed by positive and negative examples are annotated with whether they accept or reject. Two states can be merged if they are indistinguishable in the resulting transition system. This feature, together with the determinism of a DFA, are captured by transforming the problem into a k-color graph coloring problem, where k is the fixed size of the DFA that is to be identified. The resulting graph coloring problem is then encoded as a Boolean satisfiability (SAT) query. More specifically, for each labeled word, v, the SAT encoding includes (i) a variable $x_{v,i}$ indicating if word v accesses state (color) i and (ii) a variable z_i indicating whether state (color) i is accepting.

ENCODING

We extend the existing encoding to incorporate membership respecting preferences. Using these variables, a membership preserving preference, $(w, v) \in V_\prec$, can be encoded using the following constraints:

$$\forall i, j \,.\, (w \prec_\varphi v) \in V_\prec : \underbrace{(x_{w,j} \wedge x_{v,i})}_{v \ \& \ w \text{ access } i \ \& \ j} \implies \underbrace{(z_j \implies z_i)}_{z_j \implies z_i \text{ is equiv to } z_j \leq z_i} \tag{4}$$

This constraint formalizes the previously mentioned logic that non-preferred trajectories' acceptance will lead to preferred trajectories' acceptance, and preferred trajectories' rejection will lead to non-preferred trajectories' rejection. If this were not the case, the MemReP condition would lead to a contradiction.

B. MemRePs for Monotone Predicate Families

Recall the motivating example mentioned in section 8 that we cast as a 2-dimensional monotone predicate family. We provide additional insight regarding our algorithm in the context of monotone predicate families as follows.

Geometric Perspective In order to study the generic behavior of our algorithm on monotone predicate families, it helps to focus on a geometric interpretation of the concept class. For that, we assume that we adequately parameterize the concepts in a concept class $\Phi = \{\varphi_\theta \mid \theta \in [0,1]^d\}$. The parameters θ induce a natural pointwise (or product) order $<$ with $\theta < \theta'$ if for all $i < d \,.\, \theta_i < \theta'_i$. It is then straightforward to define the order on the concepts as $\varphi_\theta \sqsubset \varphi_{\theta'}$ if $\theta < \theta'$. Next, observe that any atom, $x \in \mathcal{U}$, partitions the parameter space into two regions,

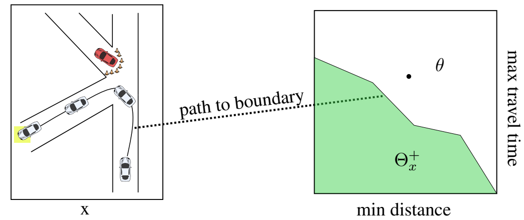

Figure 5. Mapping Ex. 8 to geometric perspective of concept class.

$$\Theta_x^+ \stackrel{\text{def}}{=} \{\theta \mid x \in \varphi_\theta\} \qquad \Theta_x^- \stackrel{\text{def}}{=} [0,1]^d \setminus \Theta^+, \tag{5}$$

called a *monotone bi-partition*. With this perspective, membership is entirely determined by whether or not the underlying concept's parameter, θ, is in the accepting set,

$$x \in \varphi_\theta \iff \theta \in \Theta^+. \tag{6}$$

Example 9. For instance, scaling and reflecting the parameters, $\theta = \left(\frac{\tau}{T}, \frac{D-d}{D}\right)$, yields a monotone parametric family. Fig 5 illustrates. In particular, the ego (white) car wants to go to the yellow region within some time budget, while maintaining a minimum distance to the red parked car. This gets mapped to a bi-partition in the normalized parameter space.

Thresholded rewards Monotone predicate families include task specifications as thresholded rewards to weighted sums over multi-dimensional rewards. In particular, let $\vec{f}(i) \in [0,1]^d$ be a sequence feature vector for $i \in [1, N]$, for $N \in \mathbb{N}$. Thresholded sums of linear rewards on these feature vectors can be cast as a $d+1$ dimensional monotone predicate family. In particular, let the thresholded sums be defined as $\sum_{i=1}^N w \cdot f(i) > \delta$ for $w \in [-1,1]^d, \delta \in [-d, d]$. We may define θ using:

$$\theta_j = \begin{cases} (1 - w_j)/2 & \text{if } j \leq d \\ (\delta/d + 1)/2 & \text{if } j = d+1 \end{cases}. \tag{7}$$

Remark. If the $\mathcal{C}(x, y)$ is total, one derives a (noiseless) variant of the learning setting considered in (Xu et al., 2020).

C. Proof sketch for Proposition 2.4

Sketch. Under exp4, a series of unproductive preference queries, i.e., ones that do not change Φ^X, will exponentially increase the weight of the historical expert. Similarly, the historical expert will exponentially increase the weight of the distinguishing membership query arm. Finally, because the per round loss is bounded, there exist a lower bound on asking distinguishing membership query. By Prop 2.4 the algorithm almost surely requires finite queries. □

D. Handling Error

Our concept learning algorithm can be adapted to gracefully handle two kinds of labeling errors: (i) *Preorder violations* and (ii) *MemRep violations*. A preorder violation occurs when the underlying order relation is observed to not be transitive or reflexive. This can be visualized using a Hasse Diagram where such a violation would either correspond to a cycle or an inconsistency in the node equivalence classes. Similarly, a MemReP violation occurs when the Hasse diagram contains an edge, (x, y), where $x \prec y$, but $x \in \varphi^*$ and $y \notin \varphi^*$.

Both classes of violations are easy to detect and isolate. In an explicit Hasse diagram representation, a topological pass over the graph suffices. In the case of SAT based concept classes, e.g., our DFA learning experiment in Section 4, we simply analyze the UNSAT-core to determine which queries must be dropped to find a consistent hypothesis. In either setting, one can alert the user, allowing the violating query responses to be dropped or corrected before resuming the learning algorithm. Combining with a final conformance tester which asks additional redundant queries from a test distribution, yields a probably approximately correct concept (Kearns & Vazirani, 1994).

E. Learning DFAs: Additional Details and Results

E.1. Tomita Languages

The Tomita languages (Tomita, 1982) are a standard set of regular languages and DFAs frequently used as a benchmark in DFA learning and identification. The seven languages have a number of appealing qualities: they are relatively parsimonious, and they collectively span a number of interesting properties, including distributions of accepting and rejecting strings, existence of sink states, and relative ease of identification with a small number of membership queries.

For the Tomita languages, we used a manually designed preference ordering to incoporate more semantic meaning into the ordering and thereby encourage better learning from preference queries themselves. The preference ordering works as follows: Positively labeled atoms are still always preferred over negatively labeled atoms, as expected. When comparing two negatively labeled atom, the atom that took longer to reach a sink state (i.e., a rejecting state that cannot be transitioned out from) in the DFA was preferred, if a sink state existed. If neither atom reached a sink state, the atom with a longer accepting prefix was preferred. When comparing two positive atoms, we enumerate the four possible two-token extensions to the atoms and prefer the atom that is more frequently accepted when considering all of the extensions.

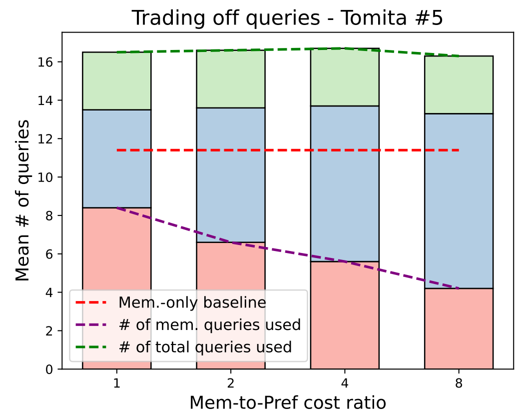

Figure 6. Trade-off between preference queries and membership queries for tomita language #5. The bars plotted show the contribution of membership (red, bottom), preference (blue, middle), and equivalence (green, top) queries.

In Tables 1, 2, and 3, we show the results for the Tomita languages experiment for our MemRePs algorithm and the membership query-only baseline (where the cost for asking a preference query is set to ∞), averaged across 20 trials. Note that we omit the comparison between the number of equivalence queries asked by each method since these numbers were the same for both. Overall, we notice that the tradeoff between membership and preference queries is apparent as the relative cost between the two increases. However, this tradeoff is more pronounced in some languages (languages #4, 5, and 7) than other (languages #1, 2, and 6). An example language (language #5) is further illustrated in

We also note that the number of preference queries needed in some cases, such as in languages 4 and 7, dramatically increase with the increased cost ratio. The high variance for preference queries needed also indicates that the combination of the CMAB algorithm and atom selection process leaves room for improvement and consistency.

E.2. Robustness Experiment

As mentioned in the main text, we designed our algorithm to be robust in noisy settings, where the response to a query is flipped to be incorrect some proportion of the time. In our algorithm's implementation for DFAs, if a labeling error occurs that causes a violation, the UNSAT-core is extracted to see which existing assumptions caused this violation, and

	DFA 1		DFA 2		DFA 3		DFA 4		DFA 5		DFA 6		DFA 7	
	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base
Cost=1	3.1	3.6	6.6	7.0	5.1	7.3	12.5	16.6	8.4	11.4	5.7	6.7	17.3	18.7
Cost=2	3.1	3.6	6.2	7.0	4.7	7.3	11.2	16.6	6.6	11.4	5.5	6.7	16.1	18.7
Cost=4	3.0	3.6	5.3	7.0	3.8	7.3	11.4	16.6	5.6	11.4	5.5	6.7	14.9	18.7
Cost=8	3.0	3.6	3.5	7.0	2.7	7.3	6.8	16.6	4.2	11.4	5.4	6.7	12.2	18.7

Table 1. Mean number of membership queries asked by our membership-and-preference selection algorithm (Ours) in comparison to the membership-only baseline (Base) on the seven Tomita DFAs. Little to no difference is seen in the simpler DFAs, whereas the discrepancy in query amount is more pronounced in more complicated DFAs.

	DFA 1		DFA 2		DFA 3		DFA 4		DFA 5		DFA 6		DFA 7	
	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base
Cost=1	0.13	0.23	0.74	0	0.66	0.49	1.57	1.61	1.28	1.01	0.67	0.50	2.00	1.78
Cost=2	0.43	0.23	1.01	0	0.61	0.49	1.62	1.61	1.18	1.01	0.92	0.50	0.87	1.78
Cost=4	0.22	0.23	0.78	0	0.87	0.49	1.58	1.61	0.64	1.01	0.81	0.50	2.19	1.78
Cost=8	0.41	0.23	0.67	0	0.64	0.49	0.97	1.61	1.10	1.01	0.49	0.50	1.30	1.78

Table 2. Variances for number of membership queries asked by our membership-and-preference selection algorithm in comparison to the membership-only baseline on the seven Tomita DFAs.

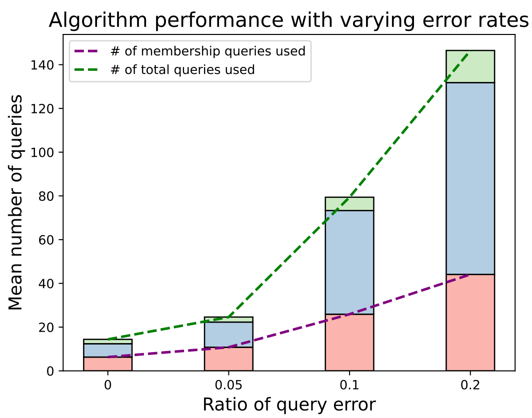

Figure 7. Greater numbers of queries are required to learn the target DFA (Tomita language #6) as the error rate increases. The bars plotted show the contribution of membership (red, bottom), preference (blue, middle), and equivalence (green, top) queries.

those assumptions are then dropped. We demonstrate the effect of labeling errors in an experiment, where an example DFA (in this case, Tomita language #6) is learned in settings of increasing proportions of labeling errors. The results are displayed in figure 7, where an increasing rate of error causes more assumptions that need to be dropped by our algorithm, resulting in more queries required to learn the correct concept. In other words, as errors are made more frequently and assumptions are discarded more often, it becomes harder for the necessary set of assumptions to be efficiently obtained by the learner.

E.3. Scalability Experiments

To understand how our algorithm scales as our target DFA increases in complexity, we evaluate our algorithm's scalability on a simple one-symbol language that determines whether the length of an input sequence is modulo some positive integer k. The size of the DFA is k states with a single accepting state. We provide the results of our algorithm's performance as k increases in Table 4. Not included in the table is the number of equivalence queries used in each DFA, which did not vary as a function of membership-to-preference cost ratio. The mean number of equivalence queries asked were 6.4, 9.1, 9.9, and 22.6 for DFA states 5, 10, 20, and 40, respectively. The equivalence queries in this setting were highly informative, allowing the number of other query types to scale efficiently but somewhat restricting those queries' utility.

In addition to the previous experiment, we generalized Tomita Language #4, which originally is defined as a 3-state DFA that encodes the task "any string without more than 2 consecutive '0's", to any string with more than n consecutive '0's. The size of the DFA in states is $n + 2$, including the rejecting sink state. We vary n from 1 to 4 and present our results in Table 5. We note that the number of queries required quickly increases with the increase in number of states, which is to be expected given the super-linear increase in search space size with number of states.

	DFA 1		DFA 2		DFA 3		DFA 4		DFA 5		DFA 6		DFA 7	
	Mean	Var.	Mean	Var.	Mean	Var.	Mean	Var.	Mean	Var.	Mean	Var.	Mean	Var.
Cost=1	1.4	0.75	4.7	1.11	4.6	2.72	10.5	6.42	5.1	2.14	3.3	1.10	11.5	4.18
Cost=2	2.6	0.88	5.2	0.92	5.0	1.66	25.1	11.1	7.0	1.76	3.3	2.28	22.0	9.27
Cost=4	3.1	0.79	6.3	0.83	9.7	2.05	41.3	12.71	8.1	2.95	3.7	1.44	44.3	8.22
Cost=8	3.8	0.83	8.1	0.77	15.9	2.28	84.3	17.23	9.1	5.11	4.1	1.34	78.7	16.65

Table 3. Mean and variance for number of preference queries asked by our membership-and-preference selection algorithm on the seven Tomita DFAs.

	5 States		10 States		20 States		40 States	
	# Mem.	# Pref.	# Mem.	# Pref.	# Mem.	# Pref.	# Mem.	# Pref.
Cost=1	2.2	1.2	5.7	3.6	9.4	7.4	18.7	7.8
Cost=2	2.0	1.5	4.7	5.6	8.1	8.4	17.9	9.9
Cost=4	1.4	2.6	4.6	6.6	6.2	12.2	15.4	19.5
Cost=8	1.0	2.9	3.9	10.9	3.7	17.8	11.3	31.4

Table 4. Number of Membership and Preference Queries for the scaled modulo DFA structure, averaged over ten trials.

	3 States		4 States		5 States		6 States	
	# Mem.	# Pref.	# Mem.	# Pref.	# Mem.	# Pref.	# Mem.	# Pref.
Cost=1	7.3	3.1	12.5	10.5	23.3	14.3	32.4	34.6
Cost=2	6.5	6.3	11.2	25.1	20.3	26.9	28.6	59.0
Cost=4	5.6	17.7	11.4	41.3	19.1	71.7	25.8	143.6
Cost=8	5.1	34.1	6.8	84.3	17.9	147.2	23.6	218.6

Table 5. Number of Membership and Preference Queries for the scaled Tomita #4 experiment, averaged over ten trials. The number of equivalence queries remained constant over costs.