Learning Quantitative Automata Modulo Theories

Eric Hsiung[1] , **Nathan Tsoi**[1] , **Swarat Chaudhuri**[1,2] and **Joydeep Biswas**[1]
[1]The University of Texas at Austin
[2]Google Deepmind
{ehsiung, nathan.tsoi, swarat, joydeepb}@cs.utexas.edu

Abstract

We introduce QUINTIC, a general algorithm for actively learning quantitative automata from preferences. Quantitative automata evaluate input sequences by applying a valuation function—such as sum, product, or average—to the output labels of the states visited. Such models naturally arise in use cases ranging from probabilistic verification, sequence classification, and sequential decision making. However, existing learning approaches, such as variants of L^*, weighted-automata learning algorithms, and active learning preference-driven methods, either assume finite output alphabets or restrict the valuation function to particular forms. QUINTIC utilizes a symbolic observation table and applies deductive reasoning with the assistance of an SMT solver to identify the correct minimal state, transition, and state label combination of the quantitative automaton. The deductive reasoning relies on the minimal combination of theories determined by the valuation function and output alphabet. Consequently, QUINTIC has completeness, minimalism, and query complexity guarantees, and learns quantitative automata across finite, integer, and rational outputs. Our extensive experiments show how QUINTIC scales under weak or strong feedback, and alternative MaxSMT objectives.

1 Introduction

Sequential decision-making problems in reinforcement learning, robotics, and planning are often driven by quantitative objectives that assign each possible behavior a value. For example, objectives can minimize costs or maximize cumulative Markovian rewards. However, non-Markovian tasks require reasoning over state-action histories rather than only current state [Abel *et al.*, 2021]. Objectives of non-Markovian tasks can be naturally encoded as *quantitative automata* [Chatterjee *et al.*, 2008]: finite-state machines that assign numeric values to input sequences by aggregating encountered output values via a *valuation function*. We aim to learn quantitative automata from preferences, as a means for agents to model preferred behaviors for solving non-Markovian tasks.

Automata learning can be categorized into passive and active learning. Passive methods learn automata from datasets of input-output examples [Oncina and García, 1992; Dupont, 1994], but such methods are limited by incomplete or uninformative data. Desired behaviors which cannot be precisely demonstrated in the given dataset will be difficult to learn.

Active approaches, such as L^*, learn by querying a teacher to gather information [Angluin, 1987; Isberner *et al.*, 2014]. For example, *membership queries* evaluate set inclusion of a sequence, while *equivalence queries* check automaton correctness. The key advantage of active learning over passive learning is the ability to select informative sequences and discover unseen transitions. Yet, most active methods burden the teacher with providing a precise output for each input.

Preference-based learning offers an alternative approach. Rather than requiring exact outputs, *preference queries* ask which of two sequences is preferred. Preference queries are easier to answer, require no exact numeric values, and naturally capture difficult-to-formalize constraints about desired system behaviors. Learning automata from preferences, however, remains relatively unexplored.

Actively learning quantitative automata from preferences introduces new challenges. Existing active preference-based algorithms assume classification-based preferences and operate over finite alphabets [Shah *et al.*, 2023; Hsiung *et al.*, 2025]. However, preferences based on aggregations of outputs (e.g. summation) often induce *ambiguity* in identifying the automaton structure, with multiple hypotheses satisfying the same preference set. Resolving the ambiguity demands a more expressive reasoning framework.

Our approach, QUINTIC (**Qu**antitative Automata **In**ference from **T**heory and **I**ncremental **C**onstraints), actively learns deterministic quantitative automata (DQAs) from preference and equivalence queries with the style of an L^*-based algorithm. QUINTIC utilizes a symbolic observation table and applies deductive reasoning with an SMT solver to identify the state, transition, and state label combination consistent with collected preferences and the valuation function. Minimalism of the result and search completeness are guaranteed by iterative deepening, with a MaxSMT objective guiding search and inference. Our empirical experiments show how query complexity and solver time scale as a function of valuation function, feedback strength, and search objective.

In summary, we contribute: (a) QUINTIC, a novel active al-

gorithm for learning DQAs from preferences, (b) guarantees of minimalism and search completeness, and (c) empirical scaling and ablation experiments. Code and supplementary material are available at https://eric-hsiung.github.io/quintic/.

2 Related Work

Active automata learning approaches commonly use *observational equivalence* to identify states. Observationally equivalent states yield identical observations. We discuss methods that learn from preferences and data. Preference queries return the relative ordering of a pair of sequences. Classification-based preferences (CbPs) order sequences by the ordering of classes to which the sequences belong. Valuation-based preferences (VbPs) order sequences according to the computed numerical value of each sequence.

Automata Learning from Preferences. State-merging has been used to learn deterministic finite automata (DFAs) from a combination of CbP and membership queries [Shah *et al.*, 2023]. DFAs are binary classifiers of sequences and represent a language. Moore machines are multiclass classifiers of sequences, generalize DFAs, and are a specific instance of DQAs. Most similar to our approach, REMAP actively learns Moore machines via CbPs queries and applies *unification* to identify states [Hsiung *et al.*, 2025]. Unification identifies sets of equivalent variables and uses representatives of those sets to rewrite representations. QUINTIC uses VbP queries, which subsume CbP queries, in order to learn the states, transitions, and state labels of DQAs. Observational equivalence becomes inherently ambiguous with VbPs, compared to CbPs. Multiple transition functions may satisfy a set of VbPs. Only one transition function satisfies a set of CbPs. REMAP disregards ambiguity; QUINTIC resolves it using deductive reasoning.

Automata Learning from Data. Numerous methods learn from queries that return numerical values. The L^* algorithm learns DFAs through membership and equivalence queries [Angluin, 1987]. DFAs have also been learned under ambiguous membership query responses. [Leucker and Neider, 2012]. Concepts introduced by L^* have been adapted for learning different types of automata using various queries. For instance, weighted automata value sequences by taking a generalized sum-of-products of transition weights [Balle and Mohri, 2015; Bergadano and Varricchio, 1994]. Actively learning weighted automata relies on value queries that return the computed numerical value of a sequence. The queries yield a system of equations with a unique solution for the transition weights [Balle and Mohri, 2015]. In QUINTIC, preference queries yield a system of inequalities, which corresponds to a set of solutions for state labels of which only a single solution is correct. Other methods use queries that return numerical values restricted to a finite output alphabet. Algorithms for learning reward machines, which are automata that emit rewards upon transitions, use queries that return reward values [Tappler *et al.*, 2019; Gaon and Brafman, 2020; Xu *et al.*, 2021; Dohmen *et al.*, 2022]. The Λ^* algorithm learns symbolic automata, which use expressions to summarize sets of transitions, from membership queries [Argyros and D'antoni, 2018]. In QUINTIC, the value of a sequence is not restricted to the output alphabet. Instead, the value is an element from the set of all possible aggregations of output alphabet values defined by the range of the valuation function.

Existing active methods cannot solve the problem of learning DQAs from VbP constraints. A membership query identifies the label of one state in the target automaton [Angluin, 1987]. CbP queries compare one pair of sequence labels at a time [Hsiung *et al.*, 2025]. However, VbP queries result in relations between *aggregations of multiple unknown values*. Therefore, existing algorithms, such as L^* and REMAP, assume that the set of states is *unique* for a given set of observations. The assumption does not hold for VbP constraints. QUINTIC handles the ambiguity arising from VbPs.

3 Problem Formulation

Consider a learner $\mathcal{L}$ that actively learns the minimal DQA $\mathcal{A}$ that exactly represents the function $\mathcal{V}: (\Sigma^I)^* \to \mathbb{R}$ from preference information. $\mathcal{V}$ maps each input sequence composed from zero or more elements from the input alphabet Σ^I to a real number. Here, $\mathcal{A}$ is $\langle Q, q_0, \Sigma^I, \Sigma^O, \delta, L, \mathbf{Val}\rangle$, with a finite set of states Q, initial state $q_0 \in Q$, input and output alphabets Σ^I and Σ^O, transition function $\delta: Q \times \Sigma^I \to Q$, and labeling function $L: Q \to \Sigma^O$ which assigns outputs to states. The valuation function $\mathbf{Val}: (\Sigma^I)^* \to \mathbb{R}$ folds state labels traversed by a sequence into a real value via an aggregation operator $\oplus: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. We define the j-length prefix of s as $s_{:j}$, the length of s as $|s|$, the concatenation of sequences s (prefix) and e (suffix) as $s \cdot e$, and an element of Σ^I as σ. The extended transition function $\hat{\delta}: Q \times (\Sigma^I)^* \to Q$ simulates a sequence of transitions with $\hat{\delta}(q_0, \sigma \cdot s) = \hat{\delta}(\delta(q_0, \sigma), s)$. Then, $\mathcal{A}(s) = \mathbf{Val}(s) = \bigoplus_{j=0}^{|s|} L\big(\hat{\delta}(q_0, s_{:j})\big)$ aggregates the traversed state labels. Figure 3a shows a summation example. Q, δ, and L are learned from VbP and equivalence queries.

VbP queries yield the relative ordering of two sequences according to $\mathcal{V}$, and *equivalence queries* indicate whether a hypothesis automaton values every sequence identically to $\mathcal{V}$. We assume the teacher has preferences consistent with $\mathcal{V}$, and responds to preference and equivalence queries as follows:

Definition 1 (Preference Query)**.** *A preference query* $\mathbf{prefQ}: (\Sigma^I)^* \times (\Sigma^I)^* \to \{1, 0, -1\}$ *is evaluated as*

$$\mathbf{prefQ}(s_1, s_2) = \begin{cases} 1 & \text{if } \mathcal{V}(s_1) > \mathcal{V}(s_2) \\ 0 & \text{if } \mathcal{V}(s_1) = \mathcal{V}(s_2) \\ -1 & \text{if } \mathcal{V}(s_1) < \mathcal{V}(s_2) \end{cases} \tag{1}$$

Definition 2 (Equivalence Query)**.** *An equivalence query returns a triple in* $\mathbb{B} \times (\Sigma^I)^* \times \mathbb{R}$*: is* $\mathcal{A}$ *correct or not; if not, then a counterexample sequence* c *where the evaluation of* c *by* $\mathcal{A}$ *and by* $\mathcal{V}$ *do not match; and a real value from feedback function* $f(c, \mathcal{A}, \mathcal{V})$ *with semantics understood by the learner.*

$$\mathbf{equivQ}(\mathcal{A}) = \begin{pmatrix} \forall s \in (\Sigma^I)^* : \mathcal{A}(s) = \mathcal{V}(s) \\ c : \mathcal{A}(c) \neq \mathcal{V}(c) \\ f(c, \mathcal{A}, \mathcal{V}) \end{pmatrix} \tag{2}$$

3.1 Preliminaries

Automata learning algorithms learn the states Q, the transition function δ, and the labeling function L for the automaton being learned. Input and output alphabets Σ^I and Σ^O are

typically known. An *alphabet* is a finite set of elements used to construct sequences. A *sequence* is a list of alphabet elements; an input sequence $\sigma_1 \cdots \sigma_k$ represents sequence of k transitions taken in an automaton.

L*, Queries, and Observational Equivalence. L*-based learners query a teacher for observations about sequences. Many query types either return the label of a sequence from the output alphabet (e.g. membership queries), or how the labels of a pair of sequences are related (e.g. CbP queries). Observations are recorded in a 2D *observation table* $\mathcal{O}$. Based on the table structure and the type of observations gathered, an explicit definition for *observational equivalence* determines if two rows of $\mathcal{O}$ represent the same state. States and transitions are identified by applying *observational equivalence*. Because $\mathcal{O}$ is used to represent Q, δ, and L of the desired automaton, certain conditions are imposed to ensure $\mathcal{O}$ meets the formal automaton definition of the desired automaton. Once this occurs, an automaton can be constructed from $\mathcal{O}$ and checked for correctness with an equivalence query to the teacher. Alternation between construction and correctness checks continues until the correct automaton is found. For an in-depth overview of L*, we refer the reader to Appendix B.

Observation Tables and Row Equivalence. An *observation table* is a 2D array with rows and columns. Each row is indexed by a prefix and each column by a suffix. For a given prefix s and suffix e, the corresponding table entry is the observed label of the sequence $s \cdot e$. A row indexed by prefix s is denoted $\boldsymbol{row}(s)$. Observational equivalence occurs when two rows are identical: if the observations for each suffix (column) are identical between two rows, those two rows are assumed to represent the same state. Therefore, observational equivalence is defined by *row equivalence*.

Symbolic Observation Table. By contrast, in a *symbolic observation table* $\mathcal{O}$, table entries are *variables* that represent state labels. The variables are related through the constraints $\mathcal{C}$ obtained from preference and equivalence queries. Formally, $\mathcal{O}$ is the tuple $\langle S, E, T; \mathcal{C}, \Gamma \rangle$, where S is the set of prefixes, E is the set of suffixes, $\mathcal{C}$ is the set of constraints, context Γ is a bijective mapping from sequences to variables, and the empirical observation function T maps each prefix-suffix sequence $s \cdot e \in (S \cup (S \cdot \Sigma^I)) \cdot E$ indexed in the table to a variable in the context Γ via the mapping $T(s \cdot e) = \Gamma[s \cdot e]$. Row equivalence is then defined *symbolically*.

Symbolic Row Equivalence. Rows $\boldsymbol{row}(s_1)$ and $\boldsymbol{row}(s_2)$ represent the same state when their entries are the same: for all $e \in E, T(s_1 \cdot e) \equiv T(s_2 \cdot e)$. Determining if two arbitrary table entries are equivalent necessitates *equivalence classes of variables* and *representatives*. An *equivalence class of variables* is a set of variables where all variables are equivalent to each other. The *representative* of an equivalence class $\mathcal{E}$ is an element of $\mathcal{E}$ used to represent all of $\mathcal{E}$ (App. C.1, Def. 13).

Bijective Construction. Constructing the automaton $\langle Q, q_0, \Sigma^I, \Sigma^O, \delta, L, \mathbf{Val} \rangle$ is as follows. First, Q is the set of rows of the table. The initial state q_0 corresponds to the empty sequence ε row:

$$Q = \{\boldsymbol{row}(s) | \text{ for all } s \in S\} \text{ with } q_0 = \boldsymbol{row}(\varepsilon) \quad (3)$$

Transitions between states are computed by appending an element σ from the input alphabet to any prefix s of the current state $\boldsymbol{row}(s)$. Then, the next state is $\boldsymbol{row}(s \cdot \sigma)$:

$$\delta(\boldsymbol{row}(s), \sigma) = \boldsymbol{row}(s \cdot \sigma) \text{ for all } s \in S \text{ and } \sigma \in \Sigma^I \quad (4)$$

Finally, the entries of the ε column are used to construct the symbolic labeling $\tilde{L}$. The labeling function L is resolved by a satisfying solution Λ to the constraint set $\mathcal{C}$.

$$\tilde{L}(\boldsymbol{row}(s)) = T(s \cdot \varepsilon) \quad (5)$$

$$\Lambda : \{T(s \cdot e) \mapsto r_{s \cdot e} | r_{s \cdot e} \in \Sigma^O\} \quad (6)$$

$$L(\boldsymbol{row}(s)) = \Lambda[T(s \cdot \varepsilon)] \quad (7)$$

A *symbolic hypothesis* is a proposed automaton that labels states with $\tilde{L}$. A *concrete hypothesis* labels states with L.

Conditions for Construction. The transition function δ for a deterministic automaton has two critical properties. It is deterministic, or *consistent*, i.e. it is a *function*. It is also *closed*, i.e. the range of δ must be a subset of the states in the domain: $\delta : Q \times \Sigma^I \to Q'$, where $Q' \subseteq Q$. Since the table $\mathcal{O}$ must represent δ over a finite set of states, $\mathcal{O}$ must also exhibit the properties of *closedness* and *consistency* prior to constructing a hypothesis automaton from $\mathcal{O}$. Both checks rely on row equivalence.

Unification Enables Row Equivalence. *Unification* is a two-step process that results in a *unified* $\mathcal{O}$. First, *equivalence classes of variables* are determined. Second, $\mathcal{O}$ is rewritten in terms of *representatives*. Unification enables closedness and consistency checks via symbolic row equivalence. Thus, constructing an automaton $\mathcal{A}$ using Equations 3-7 requires a *unified*, closed, and consistent table $\mathcal{O}$.

Equivalence Queries and Counterexample Processing. The learner always requests a check of the constructed $\mathcal{A}$ via an equivalence query. If $\mathcal{A}$ is correct, the learner terminates. If $\mathcal{A}$ is incorrect, a counterexample c is received from the teacher and incorporated into the table by the learner. L* specifically adds c and all prefixes of c to S. Alternatives exist, such as adding c and all suffixes of c to E. The purpose of expanding the table with c is to correct Q, δ, or L.

Formal Theories. A *theory* is a set of axioms and all statements which can be logically derived from those axioms.

4 QUINTIC

To learn minimal DQAs, QUINTIC (Figure 1; Algorithm 1) deductively reasons within the theory of reals about equivalence classes of variables. A class $\mathcal{E}$ is valid if it satisfies the current constraints $\mathcal{C}$ gathered in the symbolic observation table $\mathcal{O}$. Choosing a satisfying $\mathcal{E}$ allows QUINTIC to unify $\mathcal{O}$ into a unified table $\tilde{\mathcal{O}}$. However, in order to hypothesize a set of states Q, transition function δ, and symbolic labeling $\tilde{L}$, the unified table $\tilde{\mathcal{O}}$ must be closed and consistent. If $\tilde{\mathcal{O}}$ fails to be closed and consistent, then the table must be expanded and an additional $\mathcal{E}$ must be chosen. Because $\mathcal{C}$ may permit multiple satisfying solutions for $\mathcal{E}$ that remain unverifiable until an equivalence query is executed, each choice of $\mathcal{E}$ is considered a *conjecture*. The process of alternating between choosing an $\mathcal{E}$ and expanding $\mathcal{O}$ generates a *conjecture-expansion chain* (CE chain). Hence, identifying the minimal DQA is an uninformed search over CE chains that terminates once an $\mathcal{O}$ is found corresponding to the minimal DQA.

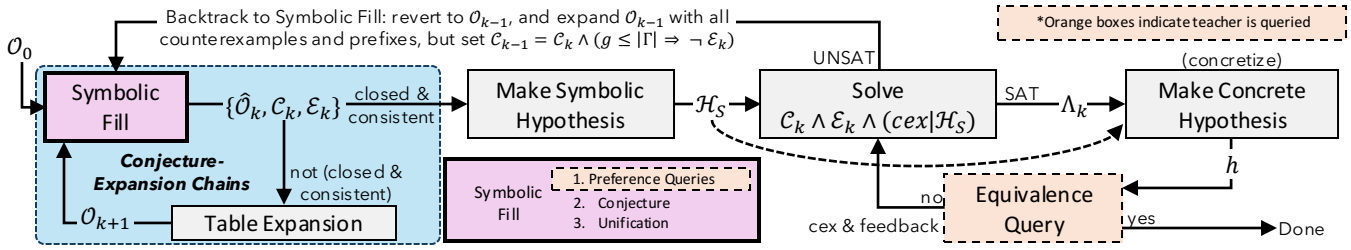

Figure 1: Diagram of QUINTIC. $\mathcal{O}_k$ and $\tilde{\mathcal{O}}_k$ are the kth *non-unified* and *unified* symbolic observation tables, respectively; $\mathcal{C}_k$ is the set of constraints from preference queries to achieve $\tilde{\mathcal{O}}_k$; the set of variable equivalence classes $\mathcal{E}_k$ resulting from conjectured variable equivalences. Starting with an initial $\mathcal{O}_0$, alternate between SYMBOLICFILLs and table expansions, progressing along a *conjecture-expansion chain* until a unified, closed, and consistent table $\tilde{\mathcal{O}}_k$ is obtained, enabling a *symbolic hypothesis* $\mathcal{H}_S$ to be constructed. A satisfying solution Λ_k to $\mathcal{C}_k$ generates a *concrete hypothesis* h which is checked via equivalence query; if h is correct, the algorithm terminates; otherwise a counterexample with feedback is returned and recorded as an extra constraint to be satisfied, represented as $(cex|\mathcal{H}_S)$. If no more satisfying solutions exist, backtrack by reverting to $\mathcal{O}_{k-1}$, then add all counterexamples and their prefixes to $\mathcal{O}_{k-1}$, while also negating $\mathcal{E}_k$. The context of sequence-to-variable mappings is Γ; $|\Gamma|$ represents the number of variables in the $\mathcal{O}_k$ table. Integer g represents a required number of variables.

Conjecture-Expansion Chains

CE chains are produced by alternating between SYMBOLICFILL (Equation 8) and TABLEEXPANSION (Equation 9) as shown in Figure 1 and lines 3-6 and 20-24 of Algorithm 1:

$$\left.\begin{array}{rcl} \mathcal{C}_{k-1} & = & \textsc{ObtainConstraints}(\mathcal{O}_{k-1}) \\ \mathcal{E}_k & = & \textsc{Conjecture}(\mathcal{O}_{k-1}, \mathcal{C}_{k-1}) \\ \tilde{\mathcal{O}}_k & = & \textsc{Unification}(\mathcal{E}_k, \mathcal{O}_{k-1}) \end{array}\right\} \quad (8)$$

$$\mathcal{O}_k = \textsc{ExpandIfNotClosedConsistent}(\tilde{\mathcal{O}}_k) \quad (9)$$

$\langle(\mathcal{E}_k, \mathcal{O}_k)\rangle_{0 \leq k \leq d}$ denotes a $(d{+}1)$-length CE chain. Only when $\mathcal{O}_d = \tilde{\mathcal{O}}_d$ can a hypothesis automaton be constructed, verified, and an incorrect conjecture be detected. Figure 2 illustrates a tree of possible chains, and Figure 3f illustrates the first few tables of each chain based on the applied conjecture.

Incorrect conjectures are handled through iterative deepening depth first search (IDS) and backtracking. By budgeting the number of times unification occurs, IDS avoids infinite length CE chains. Backtracking undoes table expansions from incorrect conjectures: if the conjunction of constraints, counterexample feedback, and conjectures is unsatisfiable, then the most recent conjecture $\mathcal{E}$ is negated under context Γ with at most m variables via logical implication, as shown on the backtracking arrow in Figure 1 and lines 12-13 of Algorithm 1:

$$|\Gamma| \leq m \implies \neg\mathcal{E} \iff \mathcal{E} \implies |\Gamma| > m \quad (10)$$

If all $\mathcal{E}$ have been ruled out under m variables, the table $\mathcal{O}$ must be expanded to include more than m variables.

From Queries to Variable Equivalence Conjectures

To gather constraints and determine the set of conjectures to choose $\mathcal{E}$ from, preference and equivalence queries are asked and responses are formalized by semantics into **Val**-constraints (constraints written in terms of the function **Val**).

Definition 3 (Preference Query Semantics)**. prefQ** *returns an element from* $\{1, 0, -1\}$*, corresponding to the relation* $\mathbf{R} \in \{>, =, <\}$ *in the statement* $\mathcal{V}(s_1)\,\mathbf{R}\,\mathcal{V}(s_2)$*. The statement is interpreted as* $\mathcal{V}(s_1)\,\mathbf{R}\,\mathcal{V}(s_2) \implies \mathbf{Val}(s_1)\,\mathbf{R}\,\mathbf{Val}(s_2)$.

Definition 4 (Feedback Semantics)**. equivQ** *returns a feedback value* $f(c, \mathcal{A}, \mathcal{V})$*. The strength of feedback provided indicates how the feedback is interpreted. If* f *is strong, then*

$$\mathbf{Val}(c) = f(c, \mathcal{A}, \mathcal{V}), \text{ indicating } \mathbf{Val}(c) = \mathcal{V}(c) \quad (11)$$

meaning the concrete valuation of c *is* $\mathcal{V}(c)$*. If* f *is weak, then*

$$\mathbf{Val}(c) \neq f(c, \mathcal{A}, \mathcal{V}), \text{ indicating } \mathbf{Val}(c) \neq \mathcal{A}(c) \quad (12)$$

meaning the concrete valuation of c *is not* $\mathcal{A}(c)$.

A recursive definition of **Val** enables **Val**-constraints to be converted into T-constraints (constraints written in terms of table variables). We consider four valuation functions with *recursive formulations*: summation ($\mathbf{\Sigma}$), discounted summation ($\gamma\mathbf{\Sigma}$), product ($\mathbf{\Pi}$), and classification ($\mathbb{N}$) (definitions in App. C.2). The recursive form for $\mathbf{\Sigma}$ is $\mathbf{Val}(s{\cdot}\sigma) = T(s{\cdot}\sigma) + \mathbf{Val}(s)$ with $\mathbf{Val}(\varepsilon) = T(\varepsilon) = 0$. Figure 3b shows conversion from **Val**-constraints to T-constraints in terms of table variables v_0, v_1, and v_2 with $\mathbf{\Sigma}$. **Val** leaves variables unbound; $\mathcal{A}$ binds variables to values.

Furthermore, the statement between a pair of table variables $T(s{\cdot}\sigma)\mathbf{R}T(s'{\cdot}\sigma')$ can be rewritten in terms of **Val** (Figure 3c). Specific combinations of preference queries encoded in the Figure 3d decision tree enable the relation **R** to be inferred. Variable pairs for which **R** is unique are considered *known* equivalences, while variable pairs for which **R** is not unique are considered *unknown* or ambiguous equivalences.

Selecting a $\mathcal{E}$ requires each ambiguous **R** to be individually assigned either an equality ($=$) or inequality ($\neq$). A MaxSMT

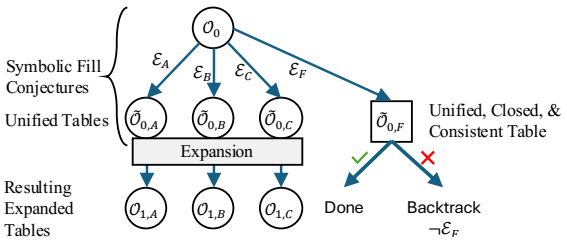

Figure 2: Diagram of Conjecture Expansion Chains. $\mathcal{O}_0$ is non-unified. $\tilde{\mathcal{O}}_{0,K}$ is unified after applying conjecture $\mathcal{E}_K$. $\mathcal{O}_{1,K}$ is the expansion step if the unified table is not closed or not consistent.

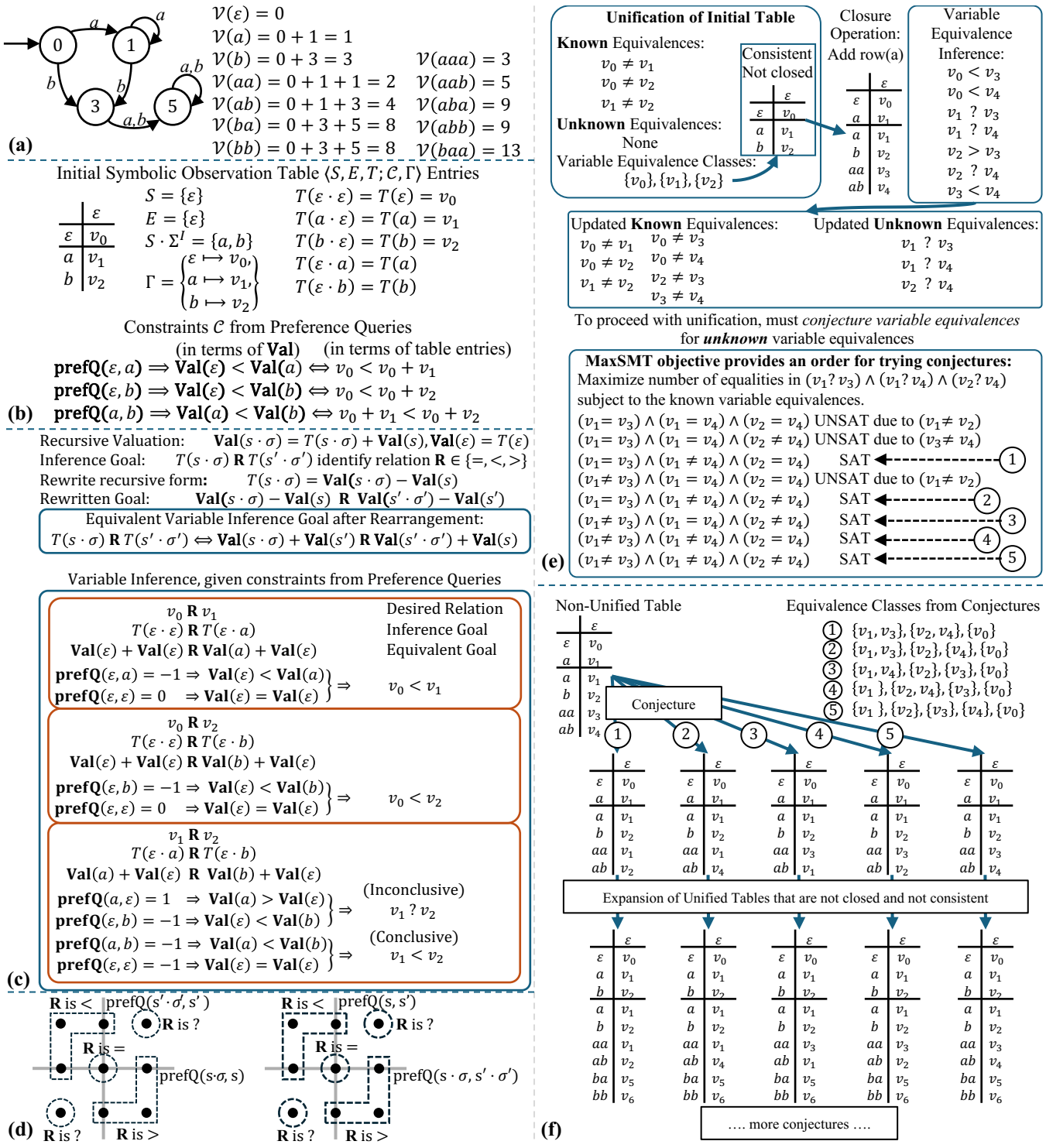

Figure 3: Example execution of QUINTIC. *(a)* summation valuation $\mathcal{V}$ represented by automaton by summing state labels, with example valuations; empty sequence ε has 0 length, $\Sigma^I = \{a, b\}$, and $\Sigma^O = \{0, 1, 3, 5\}$. *(b)* initial observation table; resulting constraints from preference queries against $\mathcal{V}$ in terms of **Val**, and table entries via recursive application of **Val**$(s \cdot \sigma) = T(s \cdot \sigma) +$ **Val**(s). *(c)* framing inferring variable equivalence, and inferred equivalences from *(b)*. *(d)* decision tree encoding inference of variable equivalence. *(e)* unification of initial table, followed by table expansion, and resulting known and unknown variable equivalences, followed by conjecturing equivalences via MaxSMT. *(f)* tree of conjecture-expansion chains and resulting tables that would result from selecting a particular variable equivalence conjecture.

Algorithm 1 QUINTIC Algorithm

1: Init $k=0$, $\mathcal{O}_k = \langle S, E, T; \mathcal{C}, \Gamma \rangle$, $S = E = \{\varepsilon\}$, $\mathcal{C} = \{\}$, $\Gamma = \emptyset$, and *cexes* $= \{\}$, *correct?* = False
2: $\tilde{\mathcal{O}}_k, \mathcal{C}_k, \mathcal{E}_k = \text{SYMBOLICFILL}(\mathcal{O}_k)$
3: **while not** $\text{ISCLOSEDANDCONSISTENT}(\tilde{\mathcal{O}}_k)$ **do**
4: $\quad \mathcal{O}_{k+1} = \text{EXPANDIFNOTCLOSEDCONSISTENT}(\tilde{\mathcal{O}}_k)$
5: $\quad \tilde{\mathcal{O}}_{k+1}, \mathcal{C}_{k+1}, \mathcal{E}_{k+1} = \text{SYMBOLICFILL}(\mathcal{O}_{k+1})$; k++
6: **end while**
7: $\mathcal{H}_S = \text{MAKESYMBOLICHYPOTHESIS}(\tilde{\mathcal{O}}_k, \Sigma^I, \Sigma^O)$
8: **while not** *correct?* **do**
9: $\quad \Lambda_k$, *sat?* $= \text{SOLVE}(\mathcal{C}_k, \mathcal{E}_k, \mathcal{H}_S, cexes)$
10: $\quad$ **if** *sat?* is UNSAT **then**
11: $\quad\quad \mathcal{O}_{k-1} = \text{PROCESSCEX}(\mathcal{O}_{k-1}, cexes)$
12: $\quad\quad \mathcal{C}_{k-1} = \mathcal{C}_k \wedge (g \leq |\Gamma| \implies \neg \mathcal{E}_k)$
13: $\quad\quad k \leftarrow k-1$, and then **goto** Line 2
14: $\quad$ **end if**
15: $\quad h = \text{MAKECONCRETEHYPOTHESIS}(\mathcal{H}_S, \Lambda_k)$
16: $\quad$ *correct?, cex, feedback* $= \text{EQUIVALENCEQUERY}(h)$
17: $\quad cexes.\text{ADD}((cex, feedback))$
18: **end while**
19: **return** h
20: **procedure** $\text{SYMBOLICFILL}(\mathcal{O}_k)$
21: $\quad \mathcal{C}_k = \text{GETCONSTRAINTSVIAPREFQUERIES}(\mathcal{O}_k)$
22: $\quad \mathcal{E}_k = \text{CONJECTUREORDERBYMAXSMT}(\mathcal{O}_k, \mathcal{C}_k)$
23: $\quad \tilde{\mathcal{O}}_k = \text{UNIFICATION}(\mathcal{E}_k, \mathcal{O}_k)$
24: $\quad$ **return** $\tilde{\mathcal{O}}_k, \mathcal{C}_k, \mathcal{E}_k$
25: **end procedure**

objective over the set of ambiguous relations prioritizes the order in which conjectures are chosen. Maximizing the number of variable equalities is a rough proxy for minimizing the number of states in the automaton. Figure 3e illustrates how an ordering over conjectures is produced by the MaxSMT objective. Other MaxSMT objectives, such as prioritizing closed and consistent tables, can also be utilized. Unification proceeds once all ambiguous $\mathbf{R}$ have been conjectured.

Counterexample Processing

Counterexample processing is identical to L*, informatively directs table expansion, and yet is unnecessary for complete search due to Equation 10 which detects required expansions. Appendix E.7 analyzes impact of counterexample processing.

5 Theoretical Guarantees of QUINTIC

Our analysis of QUINTIC considers minimalism and completeness guarantees, followed by analysis of query complexity of the algorithm. Full proofs are deferred to Appendix D.

Theorem 1 (Minimalism and Completeness)**.** *If n is the number of states of the minimal automaton isomorphic to the target, then* QUINTIC *correctly terminates after finite expansion operations with an n state automaton at the end of a CE chain containing at most $n-1$ expansion operations.*

Proof Sketch. Minimalism and completeness are a consequence of (1) the completeness of IDS, whether in a constrained or unconstrained solution space, (2) showing a bijection between partitions of conjectures and the number of states in the table, and (3) showing the correct solution must exist on a CE chain of at most length $n-1$. □

Theorem 2 (Query Complexity)**.** *If n is the number of states of the minimal automaton isomorphic to the target, and m is the maximum length of any counterexample the teacher returns, then (a)* QUINTIC *executes at most $N = n^2|\Sigma^O|^n \left\{ {|\Sigma^I|n \atop n} \right\}$ equivalence queries, where $\left\{ {r \atop t} \right\}$ is a Stirling number of the second kind (the number of ways to partition r elements into t non-empty subsets), and (b) the preference query complexity is at most $\mathcal{O}\left(|\Sigma^I|(n+mN)^3 \log(|\Sigma^I|(n+mN)^3)\right)$.*

Proof Sketch. The number of equivalence queries leverages results from Theorem 1 and Bassino and Nicaud to upper bound the number of automata $\mathcal{A}_k$ with k states [Bassino and Nicaud, 2007]; a bound for the sum of $\mathcal{A}_k$ over k from 1 to n yields the equivalence query result. Preference query complexity depends on $|(S \cup (S \cdot \Sigma^I)) \cdot E|$, which is upper bounded by table size $|E||S|(1+|\Sigma^I|)$. To obtain the preference query complexity result, analyze the maximum size of the table based on a budget of $n-1$ expansion operations. □

6 Empirical Results

Our evaluations show how (1) query complexity and (2) solver time scale as a function of feedback strength, MaxSMT objective, and valuation function. The query complexity of QUINTIC is generally larger than REMAP, but surprisingly, QUINTIC achieves better query complexity than REMAP under classification-based preferences (CbPs).

6.1 Experimental Setup

We compare QUINTIC to REMAP, a CbP learning method with strong feedback and no backtracking. Query complexity and solver time were measured for learning 25 automata. We varied feedback strength and MaxSMT objectives for QUINTIC. Each automaton (shown in Appendix E.2) has four versions, one for each valuation function investigated. Results are shown in Figure 4.

Valuation Functions, Output Alphabet, and Theories. The **Val** function and Σ^O correspond to a minimal theory QUINTIC reasons over. Theory of rationals ($\mathbf{T}_{\mathbb{Q}}$) applies to: (1) both $\mathbf{\Sigma}$ and $\gamma\mathbf{\Sigma}$ with a rational Σ^O and (2) $\mathbf{\Pi}$ with a positive rational Σ^O. Theory of total ordering applies to $\mathbb{N}$ with a finite Σ^O of classes represented by integers.

QUINTIC Variants. We evaluate 4 variants of QUINTIC based on choice of feedback and MaxSMT objective. Feedback is strong (S) (Equation 11) or weak (W) (Equation 12). For MaxSMT objectives, we maximize the number of variable equalities (VE) (Appendix C.6 Equation 29 defines the VE objective). Alternatively, we maximize VE in conjunction with an objective simulating a 1-step lookahead for closedness and consistency (CC-VE) (Appendix C.7 details the CC objective). Therefore, the four variants are *strong-variable-equality* (S-VE), *weak-variable-equality* (W-VE), *strong-closed-consistent-variable-equality* (S-CC-VE), and *weak-closed-consistent-variable-equality* (W-CC-VE).

Baseline. REMAP can only handle CbP queries. Variants S-VE and S-CC-VE under CbP are most similar to REMAP.

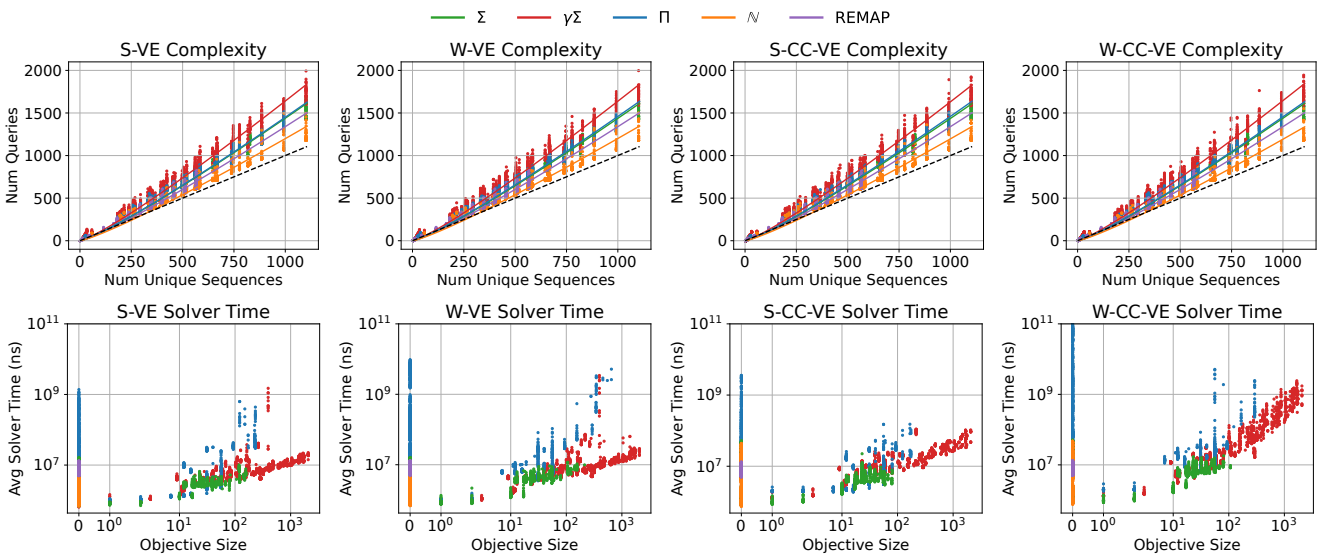

Figure 4: Query Complexity and Solver Time of QUINTIC variants under different feedback and MaxSMT objectives, as a function of valuation function. Legend: S (strong feedback), W (weak feedback), VE (single MaxSMT objective, App. Eqn. 29), CC-VE (joint MaxSMT objectives closedness and consistency objective, defined in App. Eqn. 29 and App. C.7), REMAP is the baseline. Dashed black line: L^* complexity. Top: Preference Query Complexity with best-fit $N \ln N$ curves. Bottom: Average Solver Time vs MaxSMT Objective Size.

Implementation. Each QUINTIC variation is run 100 times per automaton, for a total of 2500 trials per variant, and 10000 trials overall. Counts of preference and equivalence queries and total solver time are measured per trial. Constraints are encoded using a rational representation; Z3 is used for SMT and MaxSMT solving [De Moura and Bjørner, 2008]. Exact equivalence queries are implemented with the Hopcroft-Karp algorithm which checks for equivalence by inspecting the transition function and state labels [Hopcroft and Karp, 1971; Almeida *et al.*, 2009]. Such a check is equivalent to computing the valuations for infinitely many sequences.

6.2 Scaling Results

We measure scaling in two ways. First, we plot preference query complexity as a function of the number of unique sequences in the table. Second, we plot mean solver time versus MaxSMT objective size in order to determine how expensive finding satisfying solutions is under $\gamma\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Pi}$, and $\mathbb{N}$.

Query Complexity. The preference query complexity of each QUINTIC variant under the $\gamma\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Pi}$, and $\mathbb{N}$ valuation-based preference queries are shown in the top row of Figure 4. In order of decreasing preference query complexity, the best-fit curves are $\gamma\boldsymbol{\Sigma}$, $\boldsymbol{\Pi}$, $\boldsymbol{\Sigma}$, REMAP, and $\mathbb{N}$, with a small difference between $\boldsymbol{\Pi}$ and $\boldsymbol{\Sigma}$. The order of curves for REMAP and $\mathbb{N}$ appears to contradict the notion that REMAP represents a lower bound on preference query complexity. However, the maximum number of unique sequences for REMAP is roughly half the maximum of the other valuations, implying that QUINTIC requires larger tables and more queries than REMAP. Suffix set E does not need to be prefix-closed for REMAP, but QUINTIC requires E to be prefix-closed. Therefore, QUINTIC ends up with larger tables than REMAP.

Solver Time. Mean solver time is compared to MaxSMT objective size in the bottom row of Figure 4. QUINTIC solves large numbers of unknown variable relations, as measured by MaxSMT objectives involving at least 1000 terms. Learning under $\gamma\boldsymbol{\Sigma}$ generates large MaxSMT objectives for some automata. Although CE chain length typically does not surpass 50 when learning the 25 automata, the total solver time can be quite expensive for certain targets, especially under $\boldsymbol{\Pi}$, weak feedback, and the CC-VE objective. In fact, the solving time depends on the valuation function used and theories employed in Z3. Solving under $\boldsymbol{\Pi}$ is the most expensive, while $\gamma\boldsymbol{\Sigma}$ generates large objectives that are still relatively quick to solve. Additional results in Appendix Figure 14 indicate QUINTIC tends to gather significantly more inequalities than REMAP. Solving time for nonlinear inequalities is expensive, as evidenced by $\boldsymbol{\Pi}$. However, large numbers of linear inequalities, from $\gamma\boldsymbol{\Sigma}$, are efficiently solvable.

7 Limitations and Conclusion

QUINTIC is an L^*-based algorithm for actively learning deterministic quantitative automata from constraints obtained from preference queries and feedback. We consider how QUINTIC learns under summation, discounted summation, product, and classification valuation functions. Significantly improving query complexity of QUINTIC would allow for use cases where humans can be queried directly for preferences. Dropping exactness in favor of probably approximately correct identification would enable a tradeoff between budgeting sampling-based equivalence queries and solution error tolerance. Conjectured variable equivalences, guided by a MaxSMT objective and iterative deepening search, ensures search completeness and minimalism in QUINTIC. We investigate how QUINTIC scales under different valuation functions, feedback strength, and MaxSMT objective.

Acknowledgements

This work has taken place in the Autonomous Mobile Robotics Laboratory (AMRL) and the Trishul Lab in the University of Texas at Austin Computer Science Artificial Intelligence Lab. AMRL research is supported in part by the NSF (CAREER-2046955, IIS-2416461, CCF-2505865, DGE-2125858). Trishul research is supported in part by DARPA (HR00112320018, HR0011-24-9-0431) and the Army Research Office (W911NF2110009). The views and conclusions contained in this document are those of the authors alone.

References

[Abel *et al.*, 2021] David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. On the expressivity of markov reward. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7799–7812. Curran Associates, Inc., 2021.

[Almeida *et al.*, 2009] Marco Almeida, Nelma Moreira, and Rogério Reis. Testing the equivalence of regular languages. In *Workshop on Descriptional Complexity of Formal Systems*, 2009.

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

[Argyros and D'antoni, 2018] George Argyros and Loris D'antoni. The learnability of symbolic automata. In *International Conference on Computer Aided Verification*, 2018.

[Balle and Mohri, 2015] Borja Balle and Mehryar Mohri. Learning weighted automata. In *Conference on Algebraic Informatics*, 2015.

[Bassino and Nicaud, 2007] Frédérique Bassino and Cyril Nicaud. Enumeration and random generation of accessible automata. *Theor. Comput. Sci.*, 381(1–3):86–104, August 2007.

[Bergadano and Varricchio, 1994] Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM J. Comput.*, 25:1268–1280, 1994.

[Chatterjee *et al.*, 2008] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic*, pages 385–400, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[De Moura and Bjørner, 2008] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, page 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[Dohmen *et al.*, 2022] Taylor Dohmen, Noah Topper, George K. Atia, Andre Beckus, Ashutosh Trivedi, and Alvaro Velasquez. Inferring probabilistic reward machines from non-markovian reward signals for reinforcement learning. In Akshat Kumar, Sylvie Thiébaux, Pradeep Varakantham, and William Yeoh, editors, *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling, ICAPS 2022, Singapore (virtual), June 13-24, 2022*, pages 574–582. AAAI Press, 2022.

[Dupont, 1994] Pierre Dupont. Regular grammatical inference from positive and negative samples by genetic search: the GIG method. In Rafael C. Carrasco and José Oncina, editors, *Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21-23, 1994, Proceedings*, volume 862 of *Lecture Notes in Computer Science*, pages 236–245. Springer, 1994.

[Gaon and Brafman, 2020] Maor Gaon and Ronen I. Brafman. Reinforcement learning with non-markovian rewards. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3980–3987. AAAI Press, 2020.

[Hopcroft and Karp, 1971] John E. Hopcroft and Richard M. Karp. A linear algorithm for testing equivalence of finite automata. Defense Technical Information Center, 1971.

[Hsiung *et al.*, 2025] Eric Hsiung, Joydeep Biswas, and Swarat Chaudhuri. Automata learning from preference and equivalence queries. In Ruzica Piskac and Zvonimir Rakamarić, editors, *Computer Aided Verification*, pages 104–126, Cham, 2025. Springer Nature Switzerland.

[Isberner *et al.*, 2014] Malte Isberner, Falk Howar, and Bernhard Steffen. The ttt algorithm: A redundancy-free approach to active automata learning. In Borzoo Bonakdarpour and Scott A. Smolka, editors, *Runtime Verification*, pages 307–322, Cham, 2014. Springer International Publishing.

[Leucker and Neider, 2012] Martin Leucker and Daniel Neider. Learning minimal deterministic automata from inexperienced teachers. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*, pages 524–538, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Oncina and García, 1992] J. Oncina and P. García. *INFERRING REGULAR LANGUAGES IN POLYNOMIAL UPDATED TIME*, pages 49–61. 1992.

[Shah *et al.*, 2023] Ameesh Shah, Marcell Vazquez-Chanlatte, Sebastian Junges, and Sanjit A. Seshia. Learning formal specifications from membership and preference queries, 2023.

[Tappler *et al.*, 2019] Martin Tappler, Bernhard K Aichernig, Giovanni Bacci, Maria Eichlseder, and Kim G Larsen. L*-based learning of Markov decision processes. In *Inter-*

national Symposium on Formal Methods, pages 651–669. Springer, 2019.

[Xu *et al.*, 2021] Zhe Xu, Bo Wu, Aditya Ojha, Daniel Neider, and Ufuk Topcu. Active finite reward automaton inference and reinforcement learning using queries and counterexamples. In *Machine Learning and Knowledge Extraction: 5th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2021, Virtual Event, August 17–20, 2021, Proceedings*, page 115–135, Berlin, Heidelberg, 2021. Springer-Verlag.