Aalborg Universitet

AALBORG UNIVERSITY

DENMARK

L*-Based Learning of Markov Decision Processes

Tappler, Martin; Aichernig, Bernhard K.; Bacci, Giovanni; Eichlseder, Maria; Larsen, Kim Guldstrand

L^*-Based Learning of Markov Decision Processes

Martin Tappler[1], Bernhard K. Aichernig[1], Giovanni Bacci[3], Maria Eichlseder[2], and Kim G. Larsen[3]

[1] Institute of Software Technology, Graz University of Technology, Graz, Austria
{aichernig,martin.tappler}@ist.tugraz.at
[2] Institute of Applied Information Processing and Communications, Graz University of Technology, Graz, Austria
maria.eichlseder@iaik.tugraz.at
[3] Dept. of Computer Science, Aalborg University, Denmark
{giovbacci,kgl}@cs.aau.dk

Abstract. Automata learning techniques automatically generate system models from test observations. These techniques usually fall into two categories: passive and active. Passive learning uses a predetermined data set, e.g., system logs. In contrast, active learning actively queries the system under learning, which is considered more efficient.
An influential active learning technique is Angluin's L^* algorithm for regular languages which inspired several generalisations from DFAs to other automata-based modelling formalisms. In this work, we study L^*-based learning of Markov decision processes, first assuming an ideal setting with perfect information. Then, we relax this assumption and present a novel learning algorithm that collects information by sampling system traces via testing. Experiments with the implementation of our sampling-based algorithm suggest that it achieves better accuracy than state-of-the-art passive learning techniques with the same amount of test observations. In contrast to existing learning algorithms with predefined states, our algorithm learns the complete model structure including the states.

Keywords: model inference · active automata learning · Markov decision processes

1 Introduction

Automata learning automatically generates models from system observations such as test logs. Hence, it enables model-based verification for black-box software systems [?,?], e.g. via model checking. Automata learning techniques generally fall into two categories: passive and active learning. Passive algorithms take a given sample of system traces as input and generate models consistent with the sample. The quality and comprehensiveness of learned models therefore largely depend on the given sample. In contrast, active algorithms actively query the system under learning (SUL) to sample system traces. This enables to steer the trace generation towards parts of the SUL's state space that have not been thoroughly covered, potentially finding yet unknown aspects of the SUL.

Many active automata learning algorithms are based on Angluin's L^* algorithm [?]. It was originally proposed for learning deterministic finite automata (DFA) accepting regular languages and later applied to learn models of reactive systems, by considering system traces to form regular languages [?]. L^* has been extended to formalisms better suited for modelling reactive systems such as Mealy machines [?,?] and extended finite state-machines [?]. Most L^*-based work, however, targets deterministic models, with the exceptions of algorithms for non-deterministic Mealy machines [?] and non-deterministic input-output transition systems [?]. Both techniques are based on testing, but abstract away the observed frequency of events, thus they do not use all available information.

Here, we present an L^*-based approach for learning models of stochastic systems with transitions that happen with some probability depending on non-deterministically chosen inputs. More concretely, we learn Markov decision processes (MDPs). Such models are commonly used to model randomised distributed algorithms [?], like in the verification of protocols [?,?]. We present two learning algorithms: the first takes an ideal view assuming perfect knowledge about the exact distribution of system traces. The second algorithm relaxes this assumption, by sampling system traces to estimate their distribution. We refer to the former as *exact learning algorithm* $L^*_{\mathrm{MDP}^e}$ and to the latter as *sampling-based learning algorithm* L^*_{MDP}. We implemented L^*_{MDP} and evaluated it by comparing it to IOALERGIA [?,?], a state-of-the-art passive learning algorithm for MDPs. Experiments showed favourable performance of L^*_{MDP}, i.e. it produced more accurate models than IOALERGIA given approximately the same amount of data. Generally, models learned by L^*_{MDP} converge in the limit to an MDP observationally equivalent to the SUL. To the best of our knowledge, L^*_{MDP} is the first L^*-based learning algorithm for MDPs that can be implemented via testing. Our contributions span the algorithmic development of learning algorithms, the implementation and the evaluation of learning algorithms. The full technical report on L^*_{MDP} [?] additionally includes convergence proofs, further experiments and implementation details.

The rest of this paper is structured as follows. We introduce notational conventions, preliminaries on MDPs and active automata learning in Sect. 2. Section 3 discusses semantics of MDPs and presents the exact learning algorithm $L^*_{\mathrm{MDP}^e}$. Section 4 describes the sampling-based L^*_{MDP}. Section 5 discusses the evaluation and in Sect. 6, we discuss related work in automata learning. We provide a summary and concluding remarks in Sect. 7.

2   Preliminaries

Notation & Auxiliary Definitions. Let S be a set. We denote the concatenation of two sequences s and s' in S^* by $s \cdot s'$, the length of a sequence s by $|s|$ and the empty sequence by ϵ. We implicitly lift elements in S to sequences of length one. Sequence s is a prefix of s' if there exists an s'' such that $s \cdot s'' = s'$, denoted by $s \ll s'$. The pairwise concatenation of sets of sequences $A, B \subseteq S^*$ is $A \cdot B = \{a \cdot b \mid a \in A, b \in B\}$. A set of sequences $A \subseteq S^*$ is prefix-closed, iff for

every $a \in A$, A also contains all prefixes of A. Suffixes and suffix-closedness are defined analogously. For a sequence s in S^*, $s[i]$ is the element at index i, with indexes starting at 1, $s[\ll i]$ is the prefix of s with length i and $prefixes(s) = \{s' \mid s' \in S^* : s' \ll s\}$ is the set of all prefixes of s. Given a multiset $\mathcal{S}$, we denote the multiplicity of x in $\mathcal{S}$ by $\mathcal{S}(x)$. $Dist(S)$ denotes the set of probability distributions over S, i.e. for all $\mu : S \to [0,1]$ in $Dist(S)$ we have $\sum_{s \in S} \mu(s) = 1$. In the remainder of this paper, distributions μ may be partial functions, in which case we implicitly set $\mu(e) = 0$ if μ is not defined for e. For $A \subseteq S$, $\mathbf{1}_A$ denotes the indicator function of A, i.e. $\mathbf{1}_A(e) = 1$ if $e \in A$ and $\mathbf{1}_A(e) = 0$ otherwise.

Markov Decision Processes.

Definition 1 (Markov decision process (MDP)). *A labelled Markov decision process (MDP) is a tuple $\mathcal{M} = \langle Q, \Sigma^{\mathrm{I}}, \Sigma^{\mathrm{O}}, q_0, \delta, L \rangle$ where Q is a finite non-empty set of states, Σ^{I} and Σ^{O} are finite sets of inputs and outputs, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma^{\mathrm{I}} \to Dist(Q)$ is the probabilistic transition function, and $L : Q \to \Sigma^{\mathrm{O}}$ is the labelling function. An MDP is* deterministic *if $\forall q \in Q, \forall i : \delta(q,i)(q') > 0 \wedge \delta(q,i)(q'') > 0 \to q' = q'' \vee L(q') \neq L(q'')$.*

We learn deterministic labelled MDPs as learned by passive learning techniques like IoAlergia [?]. Such MDPs define at most one successor state for each source state and input-output pair. In the fol-

Fig. 1. MDP model of a faulty coffee machine

lowing, we refer to these models uniformly as MDPs. We use $\Delta : Q \times \Sigma^{\mathrm{I}} \times \Sigma^{\mathrm{O}} \to Q \cup \{\bot\}$ to compute successor states. The function is defined by $\Delta(q,i,o) = q' \in Q$ with $L(q') = o$ and $\delta(q,i)(q') > 0$ if there exists such a q', otherwise Δ returns $\bot$. Fig. 1 shows an MDP model of a faulty coffee machine [?]. Outputs in curly braces label states and inputs with corresponding probabilities label edges. After providing the inputs `coin` and `but`, the coffee machine MDP produces the output `coffee` with probability 0.8, but with probability 0.2, it resets itself, producing the output `init`.
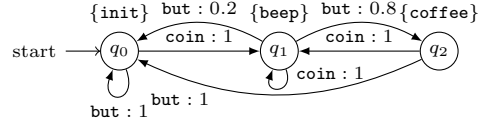
Execution. A path ρ through an MDP is an alternating sequence of states and inputs starting in the initial state q_0, i.e. $\rho = q_0 \cdot i_1 \cdot q_1 \cdot i_2 \cdot q_2 \cdots i_{n-1} \cdot q_{n-1} \cdot i_n \cdot q_n$. In each state q_k, the next input i_{k+1} is chosen non-deterministically and based on that, the next state q_{k+1} is chosen probabilistically according to $\delta(q_k, i_{k+1})$. The execution of an MDP is controlled by a so-called scheduler, resolving the non-deterministic choice of inputs by specifying a distribution over the next input given the current execution path. The composition of an MDP and a scheduler induces a Markov chain with a corresponding probability measure, see e.g. [?].

Sequences of Observations. During the execution of a finite path ρ, we observe a trace $L(\rho) = t$, i.e. an alternating sequence of inputs and outputs starting with an output, with $t = o_0 i_1 o_1 \cdots i_{n-1} o_{n-1} i_n o_n$ and $L(q_i) = o_i$. Since we consider deterministic MDPs, L is invertible, thus each trace in $\Sigma^{\mathrm{O}} \times (\Sigma^{\mathrm{I}} \times \Sigma^{\mathrm{O}})^*$ corresponds to at most one path. We say that a trace t is *observable* if there exists

a ρ with $L(\rho) = t$. In a deterministic MDP $\mathcal{M}$, each observable trace t uniquely defines a state of $\mathcal{M}$ reached by executing t from the initial state q_0. We compute this state by $\delta^*(t) = \delta^*(q_0, t)$ defined by $\delta^*(q, L(q)) = q$ and

$$\delta^*(q, o_0 i_1 o_1 \cdots i_{n-1} o_{n-1} i_n o_n) = \Delta(\delta^*(q, o_0 i_1 o_1 \cdots i_{n-1} o_{n-1}), i_n, o_n).$$

If t is not observable, then there is no path ρ with $t = L(\rho)$, denoted by $\delta^*(t) = \bot$. We denote the last output o_n of a trace $t = o_0 \cdots i_n o_n$, by $last(t)$.

We use three types of observation sequences with short-hand notations:
- *Traces:* abbreviated by $\mathcal{TR} = \Sigma^{\mathrm{O}} \times (\Sigma^{\mathrm{I}} \times \Sigma^{\mathrm{O}})^*$
- *Test sequences:* abbreviated by $\mathcal{TS} = (\Sigma^{\mathrm{O}} \times \Sigma^{\mathrm{I}})^*$
- *Continuation sequences:* abbreviated by $\mathcal{CS} = \Sigma^{\mathrm{I}} \times \mathcal{TS}$

These sequence types alternate between inputs and outputs, thus they are related among each other. In slight abuse of notation, we use $A \times B$ and $A \cdot B$ interchangeably for the remainder of this paper. Furthermore, we extend the sequence notations and the notion of prefixes to Σ^{O}, Σ^{I}, $\mathcal{TR}$, $\mathcal{TS}$ and $\mathcal{CS}$, e.g., test sequences and traces are related by $\mathcal{TR} = \mathcal{TS} \cdot \Sigma^{\mathrm{O}}$.

As noted, a trace in $\mathcal{TR}$ leads to a unique state of an MDP $\mathcal{M}$. A test sequence in $s \in \mathcal{TS}$ of length $n + 1$ consists of a trace in $t \in \mathcal{TR}$ with n outputs and an input $i \in \Sigma^{\mathrm{I}}$ with $s = t \cdot i$; thus executing test sequence $s = t \cdot i$ puts $\mathcal{M}$ into the state reached by t and tests $\mathcal{M}$'s reaction to i. Extending the notion of observability, we say that the test sequence s is observable if t is observable. A continuation sequence $c \in \mathcal{CS}$ begins and ends with an input, i.e. concatenating a trace $t \in \mathcal{TR}$ and c creates a test sequence $t \cdot c$ in $\mathcal{TS}$. Informally, continuation sequences test $\mathcal{M}$'s reaction in response to multiple consecutive inputs.

Active Automata Learning. We consider active automata learning in the minimally adequate teacher (MAT) framework [?], introduced by Angluin for the L^* algorithm. It assumes the existence of a MAT, which is able to answer queries. L^* learns a DFA representing an unknown regular language L over some alphabet A and therefore requires two types of queries: *membership* and *equivalence* queries. First, L^* repeatedly selects strings in A^* and checks if they are in L via *membership* queries. Once the algorithm has gained sufficient information, it forms a hypothesis DFA consistent with the membership query results. It then poses an *equivalence* query checking for equivalence between L and the language accepted by the hypothesis. The teacher responds either with *yes* signalling equivalence; or with a counterexample to equivalence, i.e. a string in the symmetric difference between L and the language accepted by the hypothesis. After processing a counterexample, L^* starts a new round of learning, consisting of membership queries and a concluding equivalence query. Once an equivalence query returns *yes*, learning stops with the final hypothesis as output.

L^* has been extended to learn models of reactive systems such as Mealy machines [?]. In practice, queries for learning models of black-box systems are usually implemented via testing [?]. Therefore, equivalence queries are generally only approximated as complete testing for black-box systems is impossible unless there is an upper bound on the number of system states.

3   Exact Learning of MDPs

This section presents $L^*_{\mathrm{MDP}^e}$, an exact active learning algorithm for MDPs, the basis for the sampling-based algorithm presented in Sect. 4. In contrast to sampling, $L^*_{\mathrm{MDP}^e}$ assumes the existence of a teacher with perfect knowledge about the SUL that is able to answer two types of queries: *output distribution* queries and *equivalence* queries. The former asks for the exact distribution of outputs following a test sequence in the SUL. The latter takes a hypothesis MDP as input and responds either with *yes* iff the hypothesis is observationally equivalent to the SUL or with a counterexample to equivalence. A counterexample is a test sequence leading to different output distributions in hypothesis and SUL. First, we describe how we capture the semantics of MDPs.

Semantics of MDPs. We can interpret an MDP as a function $M : \mathcal{TS} \rightarrow Dist(\Sigma^{\mathrm{O}}) \cup \{\bot\}$, mapping test sequences s to output distributions or undefined behaviour for non-observable s. This follows the interpretation of Mealy machines as functions from input sequences to outputs [**?**]. Viewing MDPs as reactive systems, we consider two MDPs to be equivalent, if their semantics are equal, i.e. we make the same observations on both.

Definition 2 (MDP Semantics). *An MDP* $\langle Q, \Sigma^{\mathrm{I}}, \Sigma^{\mathrm{O}}, q_0, \delta, L \rangle$ *induces a function M for $i \in \Sigma^{\mathrm{I}}$, $o \in \Sigma^{\mathrm{O}}$, $t \in \mathcal{TR}$:*

$$M(\epsilon)(L(q_0)) = 1$$
$$M(t \cdot i) = \bot \ \text{if} \ \delta^*(t) = \bot$$
$$M(t \cdot i)(o) = p \ \text{otherwise if} \ \delta(\delta^*(t), i)(q) = p > 0 \wedge L(q) = o$$

MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ with semantics M_1 and M_2 are output-distribution equivalent, denoted $\mathcal{M}_1 \equiv_{\mathrm{od}} \mathcal{M}_2$, iff $M_1 = M_2$.

Definition 3 (M-Equivalence of Traces). *Two traces $t_1, t_2 \in \mathcal{TR}$ are equivalent with respect to $M : \mathcal{TS} \rightarrow Dist(\Sigma^{\mathrm{O}}) \cup \{\bot\}$, denoted $t_1 \equiv_M t_2$, iff $last(t_1) = last(t_2)$ and for all continuations $v \in \mathcal{CS}$ it holds that $M(t_1 \cdot v) = M(t_2 \cdot v)$.*

A function M defines an equivalence relation on traces. Two traces are M-equivalent if they end in the same output and if their behaviour in response to future inputs is the same. Two traces leading to the same MDP state are in the same equivalence class of $\equiv_M$, as in Mealy machines [**?**]. In the following, we will refer to a function M induced by an MDP $\mathcal{M}$ also as the semantics of $\mathcal{M}$.

Queries. We are now able to define queries focusing on the observable behaviour of MDPs. Assume that we want to learn a model of a black-box deterministic MDP $\mathcal{M}$, with semantics M. Output distribution queries (**odq**) and equivalence queries (**eq**) are then defined as follows:
- *output distribution (**odq**)*: an **odq**(s) returns $M(s)$ for input $s \in \mathcal{TS}$.
- *equivalence (**eq**)*: an **eq** query takes a hypothesis Mealy machine $\mathcal{H}$ with semantics H as input and returns *yes* if $\mathcal{H} \equiv_{\mathrm{od}} \mathcal{M}$; otherwise it returns an $s \in \mathcal{TS}$ such that $H(s) \neq M(s)$ and $M(s) \neq \bot$.

Table 1. Parts of observation table for learning the faulty coffee machine (Fig. 1).

		but	coin
	`init`	$\{\texttt{init} \mapsto 1\}$	$\{\texttt{beep} \mapsto 1\}$
S	`init · coin · beep`	$\{\texttt{coffee} \mapsto 0.8, \texttt{init} \mapsto 0.2\}$	$\{\texttt{beep} \mapsto 1\}$
	`init · coin · beep · but · coffee`	$\{\texttt{init} \mapsto 1\}$	$\{\texttt{beep} \mapsto 1\}$
$Lt(S)$	`init · but · init`	$\{\texttt{init} \mapsto 1\}$	$\{\texttt{beep} \mapsto 1\}$
	$\ldots$	$\ldots$	$\ldots$

Observation Tables. Like L^*, we store information in observation table triples $\langle S, E, T \rangle$, where:

- $S \subset \mathcal{TR}$ is a prefix-closed set of traces, initialised to $\{o_0\}$, with o_0 being the initial SUL output,
- $E \subset \mathcal{CS}$ is a suffix-closed set of continuation sequences, initialised to Σ^{I},
- $T : (S \cup Lt(S)) \cdot E \to Dist(\Sigma^{\mathrm{O}}) \cup \{\bot\}$ is a mapping from test sequences to output distributions or $\bot$ denoting undefined behaviour. This mapping basically stores a finite subset of M. The set $Lt(S) \subseteq S \cdot \Sigma^{\mathrm{I}} \cdot \Sigma^{\mathrm{O}}$ is given by $Lt(S) = \{s \cdot i \cdot o | s \in S, i \in \Sigma^{\mathrm{I}}, o \in \Sigma^{\mathrm{O}}, \mathbf{odq}(s \cdot i)(o) > 0\}$.

We can view an observation table as a two-dimensional array with rows labelled by traces in $S \cup Lt(S)$ and columns labelled by E. We refer to traces in S as short traces and to their extensions in $Lt(S)$ as long traces. An extension $s \cdot i \cdot o$ of a short trace s is in $Lt(S)$ if $s \cdot i \cdot o$ is observable. Analogously to traces, we refer to rows labelled by S as short rows. The table cells store the mapping defined by T. To represent rows labelled by traces s we use functions $row(s) : E \to Dist(\Sigma^{\mathrm{O}}) \cup \{\bot\}$ for $s \in S \cup Lt(S)$ with $row(s)(e) = T(s \cdot e)$. Equivalence of rows labelled by traces s_1, s_2, denoted $\mathbf{eqRow}_E(s_1, s_2)$, holds iff $row(s_1) = row(s_2) \land last(s_1) = last(s_2)$ and approximates M-equivalence $s_1 \equiv_M s_2$, by considering only continuations in E. The observation table content defines the structure of hypothesis MDPs based on the following principle: we create one state per equivalence class of $S/\mathbf{eqRow}_E$, thus we identify states with traces in S reaching them and we distinguish states by their future behaviour in response to sequences in E (as is common in active automata learning [?]). The long traces $Lt(S)$ serve to define transitions. Transition probabilities are given by the distributions in the mapping T.

Table 1 shows a part of the observation table created during learning of the coffee machine shown in Fig. 1. The set S has a trace for each state of the MDP. Note that these traces are pairwise inequivalent with respect to $\mathbf{eqRow}_E$, where $E = \Sigma^{\mathrm{I}} = \{\texttt{but}, \texttt{coin}\}$. We only show one element of $Lt(S)$, which gives rise to the self-loop in the initial state with the input `but` and probability 1.

Definition 4 (Closedness). *An observation table $\langle S, E, T \rangle$ is closed if for all $l \in Lt(S)$ there is an $s \in S$ such that $\mathbf{eqRow}_E(l, s)$.*

Definition 5 (Consistency). *An observation table $\langle S, E, T \rangle$ is consistent if for all $s_1, s_2 \in S, i \in \Sigma^{\mathrm{I}}, o \in \Sigma^{\mathrm{O}}$ such that $\mathbf{eqRow}_E(s_1, s_2)$ it holds either that (1) $T(s_1 \cdot i)(o) = 0 \land T(s_2 \cdot i)(o) = 0$ or (2) $\mathbf{eqRow}_E(s_1 \cdot i \cdot o, s_2 \cdot i \cdot o)$.*

Algorithm 1 Making an observation table closed and consistent

---

1: **function** MakeClosedAndConsistent($\langle S, E, T \rangle$)
2:     **if** $\langle S, E, T \rangle$ is not closed **then**
3:         $l \leftarrow l' \in Lt(S)$ such that $\forall s \in S : row(s) \neq row(l') \vee last(s) \neq last(l')$
4:         $S \leftarrow S \cup \{l\}$
5:     **else if** $\langle S, E, T \rangle$ is not consistent **then**
6:         **for all** $s_1, s_2 \in S$ such that $\mathbf{eqRow}_E(s_1, s_2)$ **do**
7:             **for all** $i \in \Sigma^{\mathrm{I}}, o \in \Sigma^{\mathrm{O}}$ **do**
8:                 **if** $T(s_1 \cdot i)(o) > 0$ and $\neg\mathbf{eqRow}_E(s_1 \cdot i \cdot o, s_2 \cdot i \cdot o)$ **then**
9:                     $e \leftarrow e' \in E$ such that $T(s_1 \cdot i \cdot o \cdot e') \neq T(s_2 \cdot i \cdot o \cdot e')$
10:                     $E \leftarrow E \cup \{i \cdot o \cdot e\}$
11:     **return** $\langle S, E, T \rangle$

Closedness and consistency are required to derive well-formed hypotheses, analogously to L^* [**?**]. We require closedness to create transitions for all inputs in all states and we require consistency to be able to derive deterministic hypotheses. During learning, we apply Algorithm 1 repeatedly to establish closedness and consistency of observation tables. The algorithm adds a new short trace if the table is not closed and adds a new column if the table is not consistent.

We derive a hypothesis $\mathcal{H} = \langle Q_{\mathrm{h}}, \Sigma^{\mathrm{I}}, \Sigma^{\mathrm{O}}, q_{0\mathrm{h}}, \delta_{\mathrm{h}}, L_{\mathrm{h}} \rangle$ from a closed and consistent observation table $\langle S, E, T \rangle$, denoted $\mathcal{H} = \mathrm{hyp}(S, E, T)$, as follows:

- $Q_{\mathrm{h}} = \{\langle last(s), row(s) \rangle | s \in S\}$
- $q_{0\mathrm{h}} = \langle o_0, row(o_0) \rangle$, $o_0 \in S$ is the trace consisting of the initial SUL output
- for $s \in S$, $i \in \Sigma^{\mathrm{I}}$ and $o \in \Sigma^{\mathrm{O}}$:
  $\delta_{\mathrm{h}}(\langle last(s), row(s) \rangle, i)(\langle o, row(s \cdot i \cdot o) \rangle) = p$ if $T(s \cdot i)(o) = p > 0$
- for $s \in S$: $L_{\mathrm{h}}(\langle last(s), row(s) \rangle) = last(s)$

Learning Algorithm. Algorithm 2 implements $L^*_{\mathrm{MDP}^e}$ using queries **odq** and **eq**. First, the algorithm initialises the observation tables and fills the table cells with output distribution queries (Lines 1 to 3). The main loop in Lines 4 to 15 makes the observation table closed and consistent, derives a hypothesis $\mathcal{H}$ and performs an equivalence query **eq**($\mathcal{H}$). If a counterexample *cex* is found, all its prefix traces are added as short traces to S, otherwise the final hypothesis is returned, as it is output-distribution equivalent to the SUL. Whenever the table contains empty cells, the Fill procedure assigns values to these cells via **odq**.

Theorem 1. $L^*_{\mathrm{MDP}^e}$ *terminates and learns an MDP $\mathcal{H}$ that is output-distribution equivalent to the SUL and minimal in the number of states (proof [**?**]).*

4 Learning MDPs by Sampling

The sampling-based L^*_{MDP} is based on $L^*_{\mathrm{MDP}^e}$, but samples SUL traces instead of posing exact queries. Distribution comparisons are consequently approximated through statistical tests. While using similar data structures, L^*_{MDP} has a slightly different algorithm structure allowing to stop before reaching exact equivalence.

Algorithm 2 The main algorithm implementing $L^*_{\mathrm{MDP}^e}$

Input: Σ^{I}, exact teacher capable of answering **odq** and **eq**
Output: learned model $\mathcal{H}$ (final hypothesis)
1:   $o_0 \leftarrow o$ such that $\mathbf{odq}(\epsilon)(o) = 1$
2:   $S \leftarrow \{o_0\}$, $E \leftarrow \Sigma^{\mathrm{I}}$
3:   FILL(S, E, T)
4:   **repeat**
5:      **while** $\langle S, E, T \rangle$ not closed or not consistent **do**
6:        $\langle S, E, T \rangle \leftarrow$ MAKECLOSEDANDCONSISTENT($\langle S, E, T \rangle$)
7:        FILL(S, E, T)
8:      $\mathcal{H} \leftarrow \mathrm{hyp}(S, E, T)$
9:      $eqResult \leftarrow \mathbf{eq}(\mathcal{H})$
10:     **if** $eqResult \neq yes$ **then**
11:       $cex \leftarrow eqResult$
12:       **for all** $(t \cdot i) \in prefixes(cex)$ with $i \in \Sigma^{\mathrm{I}}$ **do**
13:         $S \leftarrow S \cup \{t\}$
14:       FILL(S, E, T)
15: **until** $eqResult = yes$
16: **return** $\mathrm{hyp}(S, E, T)$
17: **procedure** FILL(S, E, T)
18:     **for all** $s \in S \cup Lt(S), e \in E$ **do**
19:       **if** $T(s \cdot e)$ undefined **then**   ▷ we have no information about $T(s \cdot e)$ yet
20:         $T(s \cdot e) \leftarrow \mathbf{odq}(s \cdot e)$

Queries. The sampling-based teacher maintains a multiset of traces $\mathcal{S}$ for the estimation of output distributions that grows during learning. It offers an equivalence query and three queries relating to output distributions and samples $\mathcal{S}$.

- *frequency* (**fq**): given a test sequence $s \in \mathcal{TS}$, $\mathbf{fq}(s) : \Sigma^{\mathrm{O}} \to \mathbb{N}_0$ are output frequencies observed after s, where $\mathbf{fq}(s)(o) = \mathcal{S}(s \cdot o)$ for $o \in \Sigma^{\mathrm{O}}$.
- *complete* (**cq**): given a test sequence $s \in \mathcal{TS}$, $\mathbf{cq}(s)$ returns **true** if sufficient information is available to estimate an output distribution from $\mathbf{fq}(s)$; returns **false** otherwise.
- *refine* (**rfq**): instructs the teacher to refine its knowledge of the SUL by testing it directed towards rarely observed samples. Traces sampled by **rfq** are added to $\mathcal{S}$, increasing the accuracy of subsequent probability estimations.
- *equivalence* (**eq**): given a hypothesis $\mathcal{H}$, **eq** tests for output-distribution equivalence between the SUL and $\mathcal{H}$; returns a counterexample from $\mathcal{TS}$ showing non-equivalence, or returns *none* if no counterexample was found.

To implement these queries, we require the ability to reset the SUL, to perform a single input on the SUL and to observe the SUL output.

4.1   Learner Implementation

Observation Table. L^*_{MDP} also uses observation tables. They carry similar information as in Sect. 3, but instead of output distributions in $Dist(\Sigma^{\mathrm{O}})$, we store integral output frequencies ($\Sigma^{\mathrm{O}} \to \mathbb{N}_0$), from which we estimate distributions.

Definition 6 (Sampling-based Observation Table). *An observation table is a tuple* $\langle S, E, \widehat{T} \rangle$, *consisting of a prefix-closed set of traces* $S \subset \mathcal{TR}$, *a suffix-closed set of continuation sequences* $E \subset \mathcal{CS}$, *and a mapping* $\widehat{T} : (S \cup Lt(S)) \cdot E \to (\Sigma^{\mathrm{O}} \to \mathbb{N}_0)$, *where* $Lt(S) = \{s \cdot i \cdot o \,|\, s \in S, i \in \Sigma^{\mathrm{I}}, o \in \Sigma^{\mathrm{O}} : \mathbf{fq}(s \cdot i)(o) > 0\}$.

Algorithm 3 Creating compatibility classes

1: **for all** $s \in S$ **do**
2: $\quad$ rank$(s) \leftarrow \sum_{i \in \Sigma^{\mathrm{I}}} \sum_{o \in \Sigma^{\mathrm{O}}} \widehat{T}(s \cdot i)(o)$
3: $unpartitioned \leftarrow S,\ R \leftarrow \emptyset$
4: **while** $unpartitioned \neq \emptyset$ **do**
5: $\quad r \leftarrow m$ where $m \in unpartitioned$ with largest rank(m)
6: $\quad R \leftarrow R \cup \{r\}$
7: $\quad cg(r) \leftarrow \{s \in unpartitioned \,|\, \mathbf{compatible}_E(s, r)\}$
8: $\quad$ **for all** $s \in cg(r)$ **do**
9: $\quad\quad rep(s) \leftarrow r$
10: $\quad unpartitioned \leftarrow unpartitioned \setminus cg(r)$

Hypothesis Construction. As in Sect. 3, observation tables need to be closed and consistent for a hypothesis to be constructed. Here, we test statistically if cells and rows are approximately equal, referred to as compatible. The statistical tests applied in Def. 7 are based on Hoeffding bounds, as in [**?**]. Def. 8 serves as basis for adapted notions of closedness and consistency.

Definition 7 (Different). *Two sequences s and s' in $\mathcal{TS}$ produce statistically different output distributions with respect to $f : \mathcal{TS} \to (\Sigma^{\mathrm{O}} \to \mathbb{N}_0)$, denoted $diff_f(s, s')$, iff (1) $\mathbf{cq}(s) \wedge \mathbf{cq}(s') \wedge n_1 > 0 \wedge n_2 > 0$ where $n_1 = \sum_{o \in \Sigma^{\mathrm{O}}} f(s)(o)$, $n_2 = \sum_{o \in \Sigma^{\mathrm{O}}} f(s')(o)$, and (2) one of the following conditions holds:*
2a. $\exists o \in \Sigma^{\mathrm{O}} : \neg(f(s)(o) > 0 \Leftrightarrow f(s')(o) > 0)$, or
*2b. $\exists o \in \Sigma^{\mathrm{O}} : \left| \frac{f(s)(o)}{n_1} - \frac{f(s')(o)}{n_2} \right| > \left(\sqrt{\frac{1}{n_1}} + \sqrt{\frac{1}{n_2}} \right) \cdot \sqrt{\frac{1}{2} \ln \frac{2}{\alpha}}$, where α specifies the confidence level $(1-\alpha)^2$ for testing each o separately based on a Hoeffding bound [**?**,**?**].*

Definition 8 (Compatible). *Two cells labelled by $c = s \cdot e$ and $c' = s' \cdot e'$ are compatible, denoted $\mathbf{compatible}(c, c')$, iff $\neg diff_{\widehat{T}}(c, c')$. Two rows labelled by s and s' are compatible, denoted $\mathbf{compatible}_E(s, s')$ iff $last(s) = last(s')$ and the cells corresponding to all $e \in E$ are compatible, i.e. $\mathbf{compatible}(s \cdot e, s' \cdot e)$.*

Compatibility Classes. In Sect. 3, we formed equivalence classes of traces with respect to $\mathbf{eqRow}_E$ creating one hypothesis state per equivalence class. Now we partition rows labelled by S based on compatibility. Compatibility given by Def. 8, however, is not an equivalence relation, as it is not transitive in general. As a result, we cannot simply create equivalence classes. We apply the heuristic implemented by Algorithm 3 to partition S.

First, we assign a rank to each trace in S. Then, we partition S by iteratively selecting the trace r with the largest rank and computing a *compatibility class* $cg(r)$ for r. The trace r is the (canonical) representative for s in $cg(r)$, which we denote by $rep(s)$ (Line 9). Each r is stored in the set of representative traces R. In contrast to equivalence classes, elements in a compatibility class need not be pairwise compatible and an s may be compatible to multiple representatives, where the unique representative $rep(s)$ of s has the largest rank.

Definition 9 (Sampling Closedness). *An observation table $\langle S, E, \widehat{T} \rangle$ is closed if for all $l \in Lt(S)$ there is a representative $s \in R$ with $\mathbf{compatible}_E(l, s)$.*

Definition 10 (Sampling Consistency). *An observation table $\langle S, E, \widehat{T} \rangle$ is consistent if for all compatible pairs of short traces s, s' in S and all input-output pairs $i \cdot o \in \Sigma^{\mathrm{I}} \cdot \Sigma^{\mathrm{O}}$, we have that (1) at least one of their extensions has not been observed yet, i.e. $\widehat{T}(s \cdot i)(o) = 0$ or $\widehat{T}(s' \cdot i)(o) = 0$, or (2) both extensions are compatible, i.e.* $\mathbf{compatible}_E(s \cdot i \cdot o, s' \cdot i \cdot o)$.

Given a closed and consistent observation table $\langle S, E, \widehat{T} \rangle$, we derive hypothesis MDP $\mathcal{H} = \mathrm{hyp}(S, E, \widehat{T})$ through the steps below. Note that extensions $s \cdot i \cdot o$ of s in S define transitions. Some extensions may have few observations, i.e. $\widehat{T}(s \cdot i)$ is low and $\mathbf{cq}(s \cdot i) = \mathbf{false}$. In case of such uncertainties, we add transitions to a special sink state labelled by chaos, an output not in the original alphabet[1]. A hypothesis is a tuple $\mathcal{H} = \langle Q_h, \Sigma^{\mathrm{I}}, \Sigma^{\mathrm{O}} \cup \{\mathrm{chaos}\}, q_{0h}, \delta_h, L_h \rangle$ where:
  - representatives for long traces $l \in Lt(S)$ are given by (see Algorithm 3):
    $rep(l) = r$ where $r \in \{r' \in R \mid \mathbf{compatible}_E(l, r')\}$ with largest rank(r)
  - $Q_h = \{\langle last(s), row(s) \rangle \mid s \in R\} \cup \{q_{\mathrm{chaos}}\}$,
    - for $q = \langle o, row(s) \rangle \in Q_h \setminus \{q_{\mathrm{chaos}}\}$: $L_h(q) = o$
    - for q_{chaos}: $L_h(q_{\mathrm{chaos}}) = \mathrm{chaos}$ and for all $i \in \Sigma^{\mathrm{I}}$: $\delta_h(q_{\mathrm{chaos}}, i)(q_{\mathrm{chaos}}) = 1$
  - $q_{0h} = \langle L(q_0), row(L(q_0)) \rangle$
  - for $q = \langle o, row(s) \rangle \in Q_h \setminus \{q_{\mathrm{chaos}}\}$ and $i \in \Sigma^{\mathrm{I}}$ (note that $\Sigma^{\mathrm{I}} \subseteq E$):
    1. If $\neg\mathbf{cq}(s \cdot i)$: $\delta(q, i)(q_{\mathrm{chaos}}) = 1$, i.e. move to chaos
    2. Otherwise estimate a distribution $\mu = \delta_h(q, i)$ over the successor states:
       for $o \in \Sigma^{\mathrm{O}}$ with $\widehat{T}(s \cdot i)(o) > 0$: $\mu(\langle o, row(rep(s \cdot i \cdot o)) \rangle) = \dfrac{\widehat{T}(s \cdot i)(o)}{\sum_{o' \in \Sigma^{\mathrm{O}}} \widehat{T}(s \cdot i)(o')}$

Updating the Observation Table. Analogously to Sect. 3, we make observation tables closed by adding new short rows and we establish consistency by adding new columns. While Algorithm 2 needs to fill the observation table after executing MakeClosedAndConsistent, this is not required in the sampling-based setting due to the adapted notions of closedness and consistency.

Trimming the Observation Table. Observation table size greatly affects learning performance, therefore it is common to avoid adding redundant information [?,?]. Due to inexact information, this is hard to apply in a stochastic setting. We instead remove rows via a function Trim, once we are certain that this does not change the hypothesis. We remove rows that are (1) not prefixes of representatives $r \in R$, (2) that are compatible to exactly one $r \in R$, and (3) that are not prefixes of counterexamples to equivalence between SUL and hypothesis.

Learning Algorithm. Algorithm 4 implements L^*_{MDP}. It first initialises an observation table $\langle S, E, \widehat{T} \rangle$ with the initial SUL output as first row and with the inputs Σ^{I} as columns (Line 1). Lines 2 to 4 perform a refine query and then update $\langle S, E, \widehat{T} \rangle$, which corresponds to output distribution queries in $L^*_{\mathrm{MDP}^e}$. Here, the teacher resamples the only known trace $L(q_0)$.

After that, we perform Lines 6 to 19 until a stopping criterion is reached. We establish closedness and consistency of $\langle S, E, \widehat{T} \rangle$ in Line 9 to build a hypothesis

---

[1] This is inspired by the introduction of chaos states in **ioco**-based learning [?].

Algorithm 4 The main algorithm implementing L^*_{MDP}

Input: sampling-based teacher capable of answering **fq**, **rfq**, **eq** and **cq**

1: $S \leftarrow \{L(q_0)\}$, $E \leftarrow \Sigma^{\mathrm{I}}$, $\widehat{T} \leftarrow \{\}$ ▷ initialise observation table
2: perform **rfq**($\langle S, E, \widehat{T}\rangle$)   ▷ sample traces for initial observation table
3: **for all** $s \in S \cup Lt(S), e \in E$ **do**
4:    | $\widehat{T}(s \cdot e) \leftarrow$ **fq**($s \cdot e$)   ▷ update observation table with frequency information
5: $round \leftarrow 0$
6: **repeat**
7:     $round \leftarrow round + 1$
8:     **while** $\langle S, E, \widehat{T}\rangle$ not closed or not consistent **do**
9:     | $\langle S, E, \widehat{T}\rangle \leftarrow$ MAKECLOSEDANDCONSISTENT($\langle S, E, \widehat{T}\rangle$)
10:     $\mathcal{H} \leftarrow$ hyp($S, E, \widehat{T}$)   ▷ create hypothesis
11:     $\langle S, E, \widehat{T}\rangle \leftarrow$ TRIM($\langle S, E, \widehat{T}\rangle, \mathcal{H}$)   ▷ remove rows that are not needed
12:     $cex \leftarrow$ **eq**($\mathcal{H}$)
13:     **if** $cex \neq none$ **then**   ▷ we found a counterexample
14:     | **for all** $t \cdot i \in prefixes(cex)$ with $i \in \Sigma^{\mathrm{I}}$ **do**
15:     | | $S \leftarrow S \cup \{t\}$ ▷ add all prefixes of the counterexample
16:     perform **rfq**($\langle S, E, \widehat{T}\rangle$)   ▷ sample traces to refine knowledge about SUL
17:     **for all** $s \in S \cup Lt(S), e \in E$ **do**
18:     | $\widehat{T}(s \cdot e) \leftarrow$ **fq**($s \cdot e$)   ▷ update observation table with frequency information
19: **until** STOP($\langle S, E, \widehat{T}\rangle, \mathcal{H}, round$)
20: **return** hyp($S, E, \widehat{T}$)   ▷ output final hypothesis

---

$\mathcal{H}$ in Line 10. After that, we remove redundant rows of the observation table via TRIM in Line 11. Then, we perform an equivalence query, testing for equivalence between SUL and $\mathcal{H}$. If we find a counterexample, we add all its prefix traces as rows to the observation table like in $L^*_{\mathrm{MDP}^e}$. Finally, we sample new system traces via **rfq** to gain more accurate information about the SUL (Lines 16 to 18). Once we stop, we output the final hypothesis.

Stopping. The exact learner $L^*_{\mathrm{MDP}^e}$ stops upon reaching equivalence to the SUL, i.e. once there is no counterexample. In the sampling-based setting, we may not find a counterexample due to inaccurate hypotheses. Our stopping criterion therefore takes uncertainty into account, which we quantify with r_{unamb}, the relative number of (unambiguous) traces in $S \cup Lt(S)$ compatible to exactly one representative in R. Additionally, we check if the chaos state is reachable.

Consequently, we stop when (1) $r_{\mathrm{unamb}} \geq t_{\mathrm{unamb}}$ where t_{unamb} is a user-defined threshold, (2) the chaos state is unreachable, and (3) at least r_{min} rounds have been executed. We also stop after a maximum of r_{max} rounds.

4.2 Teacher Implementation

Due to space constraints, we discuss each query only briefly. An accurate description can be found in the full technical report [**?**].

- *frequency (**fq**)*: returns output frequencies observed in the sampled traces $\mathcal{S}$.
- *complete (**cq**)*: complete queries are based on threshold n_c. We consider test sequences complete that have been sampled at least n_c times.
- *refine (**rfq**)*: refine queries take an observation table $\langle S, E, \widehat{T}\rangle$ and resample incomplete sequences in $(S \cup Lt(S)) \cdot E$. The parameter n_{resample} defines how often we resample.

- *equivalence (***eq***)*: we apply two strategies for equivalence queries. *First*, we test for structural equivalence between hypothesis $\mathcal{H}$ and SUL. The testing strategy inspired by [?] performs random walks on $\mathcal{H}$ and has three parameters: n_{test}, the maximum number of tests, p_{stop}, the stop probability, and p_{rand}, the probability of choosing inputs uniformly at random. *Second*, we check for conformance between the collected samples $\mathcal{S}$ and $\mathcal{H}$ via *diff*$_{\textbf{fq}}$.

Note that we return no counterexample if trivial counterexamples containing chaos are observable in the hypothesis. This prompts L^*_{MDP} to issue further *refine* queries, causing the chaos state to be unreachable eventually. Otherwise, the observation table might grow unnecessarily which is detrimental to performance.

Convergence. We have examined convergence of the sampling-based L^*_{MDP} in the limit with respect to the following setup. We configure equivalence testing such that each input is chosen uniformly at random and the length of each test is geometrically distributed. This resembles the sampling regime assumed for IoA-LERGIA [?]. Likewise, we consider a data-dependent $\alpha_n = \frac{1}{n^r}$ with $r > 2$, where n is the number of samples collected so far. Finally, we consider L^*_{MDP} without trimming of observation tables. Informally, letting the number of rounds and thus the sample size n approach infinity, we eventually learn the correct MDP.

Theorem 2. L^*_{MDP} *as configured above creates hypotheses $\mathcal{H}_n$ that are minimal in the number of states and output-distribution equivalent to the SUL in the limit (proof [?]).*

5   Experiments

We evaluate the sampling-based L^*_{MDP} and compare it to the passive IoALERGIA [?] by learning a gridworld model with both techniques. Experimental results and the implementation can be found in the evaluation material [?]. We treat the known true MDP model $\mathcal{M}$ as a black box for learning and measure similarity to this model using two criteria: (1) the discounted bisimilarity distance [?,?] between $\mathcal{M}$ and the learned MDPs and (2) the difference between probabilistic model-checking results for $\mathcal{M}$ and learned MDPs. We compute maximal probabilities of manually defined temporal properties with all models using PRISM 4.4 [?].

Measurement Setup. As in [?], we use a data-dependent $\epsilon_N = \frac{10000}{N}$ for IoALERGIA, where N is the combined length of all learning traces. This parameter serves a role analogous to the α parameter of L^*_{MDP}. In contrast, we observed that L^*_{MDP} performs better with a fixed $\alpha = 0.05$. We sample traces for IoALERGIA with a length geometrically distributed with parameter p_l and inputs chosen uniformly at random, also as in [?]. The number of traces is chosen such that IoALERGIA and L^*_{MDP} learn from approximately the same amount of data.

We implemented L^*_{MDP} and IoALERGIA in Java. Additionally, we use the MDPDIST library [?] for bisimilarity distances, adapted to labelled MDPs. We performed the experiments with a Lenovo Thinkpad T450 with 16 GB RAM, an Intel Core i7-5600U CPU with 2.6 GHz and running Xubuntu Linux 18.04.

Table 2. Results for learning the gridworld example.

	true model	L^*_{MDP}	IoAlergia
# outputs	-	3 101 959	3 103 607
# traces	-	391 530	387 746
time [s]	-	118.3770	21.4420
# states	35	35	21
$\delta_{0.9}$	-	0.1442	0.5241
$\mathbb{P}_{\max}(F^{\leq 11}(\mathrm{goal}))$	0.9622	0.9651	0.2306
$\mathbb{P}_{\max}(\neg G\ U^{\leq 14}(\mathrm{goal}))$	0.6499	0.6461	0.1577
$\mathbb{P}_{\max}(\neg S\ U^{\leq 16}(\mathrm{goal}))$	0.6912	0.6768	0.1800

First Gridworld. Models similar to our gridworld have, e.g., been considered in the context of learning control strategies [**?**]. Basically, a robot moves around in a world of tiles of different terrains. It may make errors in movement, e.g. move south west instead of south with an error probability depending on the target terrain. Our aim is to learn an environment model, i.e. a map. Fig. 2 shows our gridworld. Black tiles are walls and other

Fig. 2. The evaluation gridworld

terrains are represented by different shades of grey and letters (Sand, Mud, Grass & Concrete). A circle marks the initial location and a double circle marks a goal location. Four inputs enable movement in four directions. Observable outputs include the different terrains, walls, and a label indicating the goal. The true model of this gridworld has 35 different states.
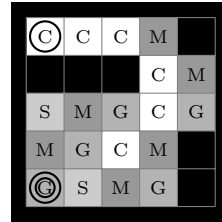
We configured sampling by $n_{\mathrm{resample}} = 300$, $n_{\mathrm{test}} = 50$, $p_{\mathrm{stop}} = 0.25$ and $p_{\mathrm{rand}} = 0.25$, and stopping by $t_{\mathrm{unamb}} = 0.99$, $r_{\mathrm{min}} = 500$ and $r_{\mathrm{max}} = 4000$. Finally, we set $p_l = 0.25$ for IoAlergia.

Results. Table 2 shows the measurement results for learning the gridworld. Our active learning stopped after 1147 rounds, sampling 391 530 traces (Row 2) with a combined number of outputs of 3 101 959 (Row 1). The bisimilarity distance discounted with $\lambda = 0.9$ to the true model is 0.144 for L^*_{MDP} and 0.524 for IoAlergia (Row 5); thus it can be assumed that model checking the L^*_{MDP} model produces more accurate results. This is indeed true for our three evaluation queries in the last three rows. These model-checking queries ask for the maximum probability (quantified over all schedulers) of reaching the goal within a varying number of steps. The first query does not restrict the terrain visited before the goal, but the second and third require to avoid G and S, respectively. The absolute difference to the true values is at most 0.015 for L^*_{MDP}, but the results for IoAlergia differ greatly from the true values. One reason is that the IoAlergia model with 21 states is significantly smaller than the minimal true model, while the L^*_{MDP} model has as many states as the true model. IoAlergia

is faster than L^*_{MDP}, which applies time-consuming computations during equivalence queries. However, the runtime of learning-specific computations is often negligible in practical applications, such as learning of protocol models [?,?], as the communication with the SUL usually dominates the overall runtime. Given the smaller bisimilarity distance and the lower difference to the true probabilities computed with PRISM, we conclude that the L^*_{MDP} model is more accurate.

Due to space constraints, we only present the intuitive gridworld experiment. The full technical report includes further experiments with a larger gridworld (72 states), a consensus protocol (272 states) and a slot machine model (109 states) [?]. They also confirm the favourable accuracy of L^*_{MDP}.

6 Related Work

In the following, we discuss techniques for learning both model structure and transition probabilities in case of probabilistic systems. There are many learning approaches for models with a given structure, e.g., for learning control strategies [?]. Covering these approaches is beyond the scope of this paper.

We build upon Angluin's L^* [?], thus our work shares similarities with other L^*-based work like active learning of Mealy machines [?]. Interpreting MDPs as functions from test sequences to output distributions is similar to the interpretation of Mealy machines as functions from input sequences to outputs [?].

Volpato and Tretmans presented an L^*-based technique for non-deterministic input-output transition systems [?]. They simultaneously learn an over- and an under-approximation of the SUL with respect to the input ouput conformance (ioco) relation [?]. Inspired by that, we apply completeness queries and we add transitions to a chaos state in case of incomplete information. Beyond that, we consider systems to behave stochastically rather than non-deterministically. Early work on ioco-based learning for non-deterministic systems has been presented by Willemse [?]. Khalili and Tacchella [?] addressed non-determinism by presenting an L^*-based algorithm for non-deterministic Mealy machines.

Most sampling-based learning algorithms for stochastic systems are passive. Notable early works are ALERGIA [?] and `rlips` [?], which identify stochastic regular languages. Both also apply Hoeffing bounds for testing for difference between probability distributions. We compare L^*_{MDP} to IOALERGIA, an extension of ALERGIA by Mao et al. [?,?]. It basically creates a tree-based representation of given system traces and repeatedly merges compatible nodes, creating an automaton. Normalised observed output frequencies estimate transition probabilities. IOALERGIA also converges in the limit. Chen and Nielsen applied it in an active setting [?], by sampling new traces to reduce uncertainty in the data. In contrast to this, we base our sampling not only on data collected so far (refine queries), but also on observation tables and derived hypothesis MDPs (refine & equivalence queries), taking information about the SUL's structure into account. In previous work, we presented a different approach to active learning

via IoAlergia which takes reachability objectives into account with the aim at maximising the probability of reaching desired events [**?**].

Feng et al. [**?**] learn assumptions for compositional verification in the form of probabilistic finite automata with an L^*-style method. Their method requires queries returning exact probabilities, hence it is not applicable in a sampling-based setting. It shares similarities with an L^*-based algorithm for learning multiplicity automata [**?**], a generalisation of deterministic automata. Further query-based learning in a probabilistic setting has been described by Tzeng [**?**]. He presented a query-based algorithm for learning probabilistic automata and an adaptation of Angluin's L^* for learning Markov chains.

7   Conclusion

We presented L^*-based learning of MDPs. For our exact learning algorithm $L^*_{\mathrm{MDP}^e}$, we assumed an ideal setting that allows to query information about the SUL with exact precision. Subsequently, we relaxed our assumptions, by approximating exact queries through sampling SUL traces via directed testing. These traces serve to infer the structure of hypothesis MDPs, to estimate transition probabilities and to check for equivalence between SUL and learned hypotheses. The resulting sampling-based L^*_{MDP} iteratively learns approximate MDPs which converge to the correct MDP in the large sample limit. We implemented L^*_{MDP} and compared it to IoAlergia [**?**], a state-of-the-art passive learning algorithm for MDPs. The evaluation showed that L^*_{MDP} is able to produce more accurate models. To the best of our knowledge, L^*_{MDP} is the first L^*-based algorithm for MDPs that can be implemented via testing. Further details regarding the implementation, convergence proofs and extended experiments can be found in the technical report [**?**] and the evaluation material [**?**].

The evaluation showed promising results, therefore we believe that our technique can greatly aid the black-box analysis of reactive systems such as communication protocols. While deterministic active automata learning has successfully been applied in this area [**?**,**?**], networked environments are prone to be affected by uncertain behaviour that can be captured by MDPs. Furthermore, L^*_{MDP} provides room for experimentation, e.g. different testing techniques could be applied in equivalence queries.