1 Approach

Equivalence queries are typically assumed to be more expensive than output queries. Many existing active learning techniques therefore focus on keeping the number of required equivalence queries low.

At a high level, Angluin's L^* algorithm for instance, can be described as follows. In each round, the algorithm first performs a sequence of output queries in a systematic way, until there is exactly one machine of minimum size that is consistent with the results from all output queries performed so far. Only then, the algorithm performs an equivalence query. If this query returns a counterexample, this implies that the correct machine must have at least one additional state. Thus, Angluin's algorithm performs at most n equivalence queries, where n is the size of the minimal correct machine.

Unlike in Angluin's setting, in general no unique machine of minimum size that is consistent with a set of observations exists. The basic idea behind our approach is to perform output queries until *all* machines of minimum size that are consistent with these queries are right-equivalent in the context of A. We then perform an equivalence query for one of these machines. If this query results in a counterexample, this counterexample witnesses that all of these machines are incorrect, and thus, the correct machine must have at least one additional state.

One challenge is to find a suitable sequence of output queries that is guaranteed to reduce the number of machines that are consistent with all queries performed so far. The basic idea is to iteratively construct all machines of minimum size that agree with all of the previous queries. We can then check whether each pair of these machines is right-equivalent. If they are not, we use a distinguishing sequence as a counterexample, without performing an equivalence query.

However, applying this approach naively would not be viable in many cases because there can be an exponential number of machines of the same size that are consistent with a set of observations, in particular in the beginning, when only a small number of queries have been performed. Thus, we identify a number of necessary conditions for candidate machines to be right-equivalent, which can be efficiently determined on observation tables. Some of these conditions correspond to notions from Angluin's algorithm, such as consistency and closedness, while others like input-completeness are special to our particular setting.

In the rest of this section, we describe our proposed algorithm in detail and introduce the necessary theoretical concepts. In particular, we describe in detail which output queries our algorithm performs to systematically reduce the number of machines that are consistent with the observations made so far. In the following, we assume that the reader is familiar with Angluin's L^* algorithm [2].

1.1 Observation tables

The main data structure used in our approach is an *observation table*. The rows of the table are indexed by a set of prefixes, the columns by a set of suffixes, and the entries of the table store the last output symbol of an output query for the concatenation of the corresponding prefix and suffix. If this concatenation is not a possible output sequence of the left machine A, we do not perform an output query, but store $\perp$ in this cell instead. In contrast to most previous definitions, our observation tables *do not* consist of two explicitly distinguished parts.

Definition 1 (Observation Table). *An observation table $T = (S, E, Q)$ consists of a finite non-empty prefix-closed set of prefixes $S \subseteq tr(A)$, a finite suffix-closed set of suffixes $E \subseteq I_B^*$ (such that $I_B \subseteq E$, and $\epsilon \notin E$), and a function $Q : (S, E) \rightarrow O_B$ such that $Q(x, e) = C_L(A^{-1}(xe))$ iff $xe \in tr(A)$ and $Q(x, e) = \perp$ otherwise.*

For a set $R \subseteq S$ and $a \in I_B$, let $Succ_T(R, a) := \{xa \mid x \in R \wedge xa \in S\}$, i.e., $Succ_T(R, a)$ is the set of successor rows for elements of R that are in the table.

In the following, we will use the term *row* both for the prefixes and for the entries of a row, when it is clear what is meant from the context.

We call two rows compatible if all columns that are not $\perp$ in both rows are the same.

Definition 2 (Compatibility). *The rows for two prefixes $x, y \in S$ are compatible iff $\forall e \in E : Q(x, e) = \perp \vee Q(y, e) = \perp \vee Q(x, e) = Q(y, e)$.*

We call an observation table consistent if whenever two rows are compatible, their successors are also compatible.

Definition 3 (Consistency). *An observation table T is consistent iff for all prefixes $x, y \in S$ such that the rows for x and y are compatible, for all $a \in I_B$ all rows in $Succ_T(\{x, y\}, a)$ are compatible.*

If there is a suffix $e \in E$ that shows that the successors of x and y under an input a are not compatible, then ae is a suffix that shows that the rows for x and y are also not compatible. Thus, we can add ae to E to resolve this inconsistency.

We define a partition of the set of rows as follows.

Definition 4 (Partition). *A partition for observation table $T = (S, E, Q)$ is a partition $P = \{P_1, ..., P_k\}$ of S, such that*

- *for all $x, y \in P_i$: the rows for x and y are compatible,*
- *for each P_i, and for all $a \in I_B$, there is a P_j, such that: $Succ_T(P_i, a) \subseteq P_j$.*

Note that if $Succ_T(P_i, a) \neq \emptyset$ then there is only one such P_j since all classes of the partition are disjoint.

We will later show how we can use partitions to build candidate machines that are consistent with the observations made so far. The words in the same class of a partition will then lead to the same states in these candidate machines.

We call a partition closed if for each class of the partition and each input symbol a, the observation table contains a successor row (under a) for at least one word of this class, if we know from the observations made so far that such a successor must exist. Our inference algorithm uses closedness as a way to determine which additional rows should be added to the table.

Definition 5 (Closedness for Partitions). *Let $P = \{P_1, ..., P_k\}$ be a partition for $T = (S, E, Q)$. P is closed if for all $P_i \in P$: if there is some $x \in P_i$ and some sequence $az \in E$ with $a \in I_B$ and $z \in I_B^*$ such that $Q(x, az) \neq \perp$, then there must be some $y \in P_i$ for which $Q(y, az) \neq \perp$, and $ya \in S$.*

Given an observation table T, let $\Pi(T, n)$ be the set of all partitions of size n. Let $\Pi_{min}(T)$ be the set of partitions of minimum size for an observation table T, i.e., $\Pi_{min}(T) = \Pi(T, m)$ where $m = min\{n \mid \Pi(T, n) \neq \emptyset\}$.

Definition 6 (Closedness). *An observation table $T = (S, E, Q)$ is closed if all minimum-size partitions $P \in \Pi_{min}(T)$ are closed.*

Definition 7 (Partial Closedness). *An observation table T is partially closed (p-closed) iff for all prefixes $x \in S$ and all sequences $az \in E$ such that $Q(x, az) \neq \perp$, there is a prefix $y \in S$ such that the rows for x and y are compatible, $Q(y, az) \neq \perp$ and $ya \in S$.*

If a table is not p-closed, then no partition can be closed.

Definition 8 (Agreement). *A Mealy machine M agrees with an observation table $T = (S, E, Q)$ if for all $x \in S$ and $e \in E$, $Q(x, e) = \perp \vee Q(x, e) = M_L(xe)$.*

For any closed partition $P = \{P_1, ..., P_k\}$ in $\Pi_{min}(T)$, we can build the following Mealy machine $M_P = (Q, I, O, \delta, q_r)$ with $k+1$ states: $Q := P \cup \{error\}$, $I := I_B$, $O := O_B \cup \perp$, $\delta(P_i, a) := (error, \perp)$ if $Succ_T(P_i, a) = \emptyset$, otherwise: $\delta(P_i, a) := (P_j, b)$ such that for some $x \in P_i$: $Q(x, a) = b \neq \perp$ and $Succ_T(P_i, a) \subseteq P_j$, and $q_r := P_i$ such that $\epsilon \in P_i$.

This machine enters a special error state if there is a class of the partition, for which the successor class is not defined.

In the following, we will use the notation $\pi_i(t)$ to denote the i-th component of a tuple t, e.g., $\pi_2(q_r, a) = a$.

Lemma 1. *Let P be a closed partition of an observation table $T = (S, E, Q)$, and $M_P = (Q, I, O, \delta, q_r)$ the Mealy machine constructed as described above. Then for all words $x \in S$, $x \in \pi_1(\delta^*(q_r, x))$.*

Theorem 1. *For a closed partition P of an observation table T, the machine M_P agrees with T.*

Definition 9. *Let $\gamma(M_P)$ be the set of machines with k states that can be obtained from M_P by removing the error state and replacing the transitions to the error state by transitions with arbitrary outputs and successor states.*

Theorem 2. *Let T be a closed observation table. Then every minimum-size machine M that agrees with T is isomorphic to an element of $\gamma(M_P)$ for some $P \in \Pi_{min}(T)$.*

Theorem 3. *If for a closed partition P the error state is not reachable in a composition of A with M_P, then all machines in $\gamma(M_P)$ are right-equivalent.*

If the error state is reachable, we can use an input sequence that leads to the error state to extend the observation table.

Definition 10 (Input-Completeness). *An observation table $T = (S, E, Q)$ is input-complete if for all minimum-size partitions $P \in \Pi_{min}(T)$, the error state is not reachable in a composition of A with M_P.*

Definition 11 (Uniqueness). *An observation table $T = (S, E, Q)$ is unique if for all pairs of minimum-size partitions $P, P' \in \Pi_{min}(T)$, the machines M_P and $M_{P'}$ are right-equivalent in the context of A.*

It follows that all machines of minimum-size size that agree with a consistent, closed, input-complete, and unique observation table are right-equivalent, and they can be obtained from the partitions.

2 Proofs

Proof of Lemma 1

Proof. By induction on the length of x.
Base Case: For $|x| = 0$, i.e., $x = \epsilon$, by construction $\epsilon \in q_r = \pi_1(\delta^*(q_r, \epsilon))$.
Induction step: Let $ya := x$ with $a \in I_B$ and $y \in I_B^*$. Since S is prefix-closed, $y \in S$. We have that $\pi_1(\delta^*(q_r, x)) = \pi_1(\delta(\pi_1(\delta^*(q_r, y)), a))$. Let $P_i := \pi_1(\delta^*(q_r, y))$. By the induction hypothesis, $y \in P_i$. Thus, $ya \in Succ_T(P_i, a)$. Since by construction $Succ_T(P_i, a) \subseteq \pi_1(\delta(P_i, a))$, $x = ya \in \pi_1(\delta(P_i, a)) = \pi_1(\delta^*(q_r, ya)) = \pi_1(\delta^*(q_r, x))$. □

Proof of Theorem 1

Proof. Let P be a closed partition of an observation table $T = (S, E, Q)$, and $M_P = (Q, I, O, \delta, q_r)$ the corresponding Mealy machine. Let $x \in S$, $e \in E$ and $Q(x, e) \neq \bot$. We will show by induction on the length of e that $Q(x, e) = M_{P_L}(xe)$.

Base Case: For $|e| = 1$, we have $M_{P_L}(xe) = \pi_2(\delta^*(q_r, xe)) = \pi_2(\delta(\pi_1(\delta^*(q_r, x)), e))$. Let $P_i := \pi_1(\delta^*(q_r, x))$. By Lemma 1, $x \in P_i$. Since $Q(x, e) \neq \bot$ and P is closed, $Succ_T(P_i, e) \neq \emptyset$. Thus, by the definition of M_P we have that $\pi_2(\delta(P_i, e)) = Q(y, e)$ for some $y \in P_i$. Since x and y are in the same class of the partition, they are compatible. Thus, $Q(y, e) = Q(x, e)$.
Induction step: Let $az := e$ with $a \in I_B$ and $z \in I_B^+$, and let $P_i \in P$ s.t. $x \in P_i$. Since P is closed, $Q(x, e) = Q(x, az) = Q(y, az)$ for some $y \in P_i$, and since $ya \in S$ and E is suffix-closed, $Q(y, az) = Q(ya, z)$. By the induction hypothesis, we have that $A(ya, z) = M_{P_L}(yaz)$. By Lemma 1, $x \in \pi_1(\delta^*(q_r, x))$ and $y \in \pi_1(\delta^*(q_r, y))$. Since the classes of P are disjoint and $x, y \in P_i$, we have that $\pi_1(\delta^*(q_r, x)) = P_i = \pi_1(\delta^*(q_r, y))$. So the inputs x and y take the machine M_P to the same state, and thus $M_{P_L}(yaz) = M_{P_L}(xaz) = M_{P_L}(xe)$. □

Proof of Theorem 2

Proof. Let $T = (S, E, Q)$ be a closed observation table. Let $M = (Q, I, O, \delta, q_r)$ be a minimum-size machine that agrees with T. We have to show that there is a $P \in \mathit{\Pi}_{min}(T)$ s.t. M is isomorphic to some machine in $\gamma(M_P)$.

Let $M' = (Q', I, O, \delta', q_r)$, with $Q' = Q \cup \{error\}$, be a machine obtained from M by replacing all transitions that are not used by any input from $(S \cup S{\cdot}E) \cap tr(A)$ with transitions to a new *error*-state with output $\perp$. This machine still agrees with T (but it has one state more than a minimum-size machine).

For each $q \in Q'$, let $P_q \subseteq S$ be the set of words s.t. M' reaches state q when reading a word from P_q, formally: $P_q = \{x \in S \mid \pi_1(\delta'^*(q_r, x)) = q\}$ (note that $P_{error} = \emptyset$).

We will now show that $P := \{P_q \mid q \in Q'\} \setminus \{P_{error}\}$ is a closed minimum-size partition for T.

- Since M' is deterministic, all elements of P are disjoint.
- The rows for all words that are in the same class of P are pairwise compatible. We will prove this by contradiction. Assume there are $x, y \in P_q$ s.t. x and y are not compatible. This means there is some $e \in E$, s.t. $Q(x, e) \neq \perp$, $Q(y, e) \neq \perp$, and $Q(x, e) \neq Q(y, e)$. But then, by the definition of agreement, $M'_L(xe) \neq M'_L(ye)$. This is a contradiction since x and y both lead to the same state $q \in Q'$ in M'.
- Let $P_q \in P$ and $a \in I_B$. We have to show that there is some $P_j \in P$ s.t. $Succ_T(P_q, a) \subseteq P_j$.

$$\begin{aligned}
Succ_T(P_q, a) &= \{xa \in S \mid x \in P_q\} \\
&= \{xa \in S \mid x \in \{y \in S \mid \pi_1(\delta'^*(q_r, y)) = q\}\} \\
&= \{xa \in S \mid \pi_1(\delta'^*(q_r, x)) = q\} \\
&\subseteq \{xa \in S \mid \pi_1(\delta'(\pi_1(\delta'^*(q_r, x)), a)) = \pi_1(\delta'(q, a))\} \\
&= \{xa \in S \mid \pi_1(\delta'^*(q_r, xa)) = \pi_1(\delta'(q, a))\} \\
&\subseteq \{x \in S \mid \pi_1(\delta'^*(q_r, x)) = \pi_1(\delta'(q, a))\} \\
&= P_{\pi_1(\delta'(q,a))}
\end{aligned}$$

  If $P_{\pi_1(\delta'(q,a))} \neq P_{error}$, then we can set $P_j := P_{\pi_1(\delta'(q,a))}$. Otherwise, $Succ_T(P_q, a) = \emptyset$, and thus any $P_j \in P$ satisfies $Succ_T(P_q, a) \subseteq P_j$.
- P is a minimum-size partition for T. We will show this by contradiction. Assume that there is a minimum-size partition P' that is smaller than P. Since T is closed, P' is closed. The machines in $\gamma(M_{P'})$ agree with T and have size $|P'| < |P| = |M|$. Contradiction.
- Since P has minimum size and T is closed, P is closed.

We will now show that the machine $M_P = (Q_P, I, O, \delta_P, q_{P,r})$ is isomorphic to M'. Let $f : Q_P \to Q'$ s.t. $f(P_q) = q$ and $f(error) = error$. f is an isomorphism between M_P and M':

- It is easy to see that f is bijective.
- $f(q_{P,r}) = q_r$, since $\epsilon \in P_{q_r}$.

- Assume that $\delta_P(P_i, a) = (error, \bot)$ for some class P_i and some input a. We have that $Succ(P_i, a) = \emptyset$. Since P is closed, there is no $x \in P_i$ s.t. $Q(x, a) \neq \bot$. Thus, there is no $x \in P_i$ s.t. $xa \in tr(A)$, and hence $\delta'(f(P_i), a) = (error, \bot)$.
- Otherwise, $\delta_P(P_i, a) = (P_j, b)$. Since for some $x \in P_i$, $Q(x, a) = b$, by agreement we have that $\pi_2(\delta'(f(P_i))) = b$. Further, since $Succ(P_i, a) \neq \bot$ there is some $x \in P_i$ s.t. $xa \in S$ and $xa \in P_j$. Thus, $\pi_1(\delta'(f(P_i))) = \pi_1(f(P_j))$.

Since M' differs from M only in the transitions that lead to the error state, M is isomorphic to some machine in $\gamma(M_P)$. □

Proof of Theorem 3

Proof. If the error state is not reachable in a composition with A, then there is no $x \in tr(A)$ s.t. M_P takes any of the transitions to the error state when reading x. Thus, modifying any of these transition does not change the behavior of the machine for inputs from $tr(A)$. Since right-equivalence only requires two machines to behave in the same way for inputs from $tr(A)$, all machines in $\gamma(M_P)$ are right-equivalent. □

3 SAT Problem

3.1 Computing the partitions

We reduce the problem of finding the partitions for a given size n, which is an NP-complete problem, to a boolean satisfiability (SAT) problem. Related reductions were used by [1] for minimizing incompletely-specified Mealy machines, and by [3] for finding DFAs that agree with a set of positive and negative input samples. However, in contrast to our approach, their approach directly computes one arbitrary machine that agrees with the samples. On the other hand, our approach computes a partition which, in general, corresponds to a machine, in which some transitions may be unspecified (if the partition is not closed or input-complete). Thus, we can exploit this additional information to determine which output queries should be performed.

SAT solvers typically require the problem to be in conjunctive normal form (CNF). We will first give a high level-description of each subproblem, and then show how to translate it to CNF.

In the following, we will use literals of the form $r_{x,i} \in \mathbb{B}$ to denote that row x is in class i of the partition. We assume that the rows are numbered from 0 to $|P|-1$.

Covering Condition: All rows of the table must be in at least one class. We therefore add, for all rows x, a clause of the form $r_{x,0} \vee r_{x,1} \vee \cdots \vee r_{x,n-1}$.

Disjointness: No row must be in more than one class. This can be expressed by adding, for all rows x and classes i, the following implication: $r_{x,i} \implies \bigwedge_{j>i} \neg r_{x,j}$. In CNF, this corresponds to the following set of clauses: $\bigwedge_{j>i}(\neg r_{x,i} \vee \neg r_{x,j})$.

Compatibility: All rows that are in the same class must be pairwise compatible. For a row x, let $Inc(x)$ be the set of states that are incompatible to x. To ensure that no incompatible rows are in the same class, we add for each row x and each class i the implication $r_{x,i} \implies \bigwedge_{\substack{y \in Inc(x) \\ y>x}} \neg r_{y,i}$. In CNF, this corresponds to $\bigwedge_{\substack{y \in Inc(x) \\ y>x}} (\neg r_{x,i} \vee \neg r_{y,i})$.

Common successors: For all rows that are in the same class i, and for which the observation table also contains their successor rows for a given input a, there must be a class j that contains all these successor rows. Thus, we have for each input symbol a and each class i a clause of the form $\exists j : \forall x : (r_{x,i} \implies r_{x',j})$, where x' is used to denote the successor row of x under input a. If the observation table does not contain this successor row, the corresponding implication is omitted.

In SAT, we can represent the existential quantifier as a disjunction, and the universal quantifier as a conjunction. The formula is thus equivalent to $\bigvee_{0 \le j < n} \quad (\bigwedge_x (\neg r_{x,i} \vee r_{x',j}))$. A direct conversion of this formula to CNF would lead to an exponential increase in its size. To obtain a more compact representation, we introduce an auxiliary literal for each input symbol and class of the form $Z_j^{a,i}$. The following formula is then equisatisfiable to the formula above: $(\bigvee_{0\leq j<n} Z_j^{a,i}) \wedge \bigwedge_{0\leq j<n} \bigwedge_x (\neg Z_j^{a,i} \vee \neg r_{x,i} \vee r_{x',j})$.

Finding a partial solution: Since we are only interested in sets of classes, the ordering of the classes does not matter. Thus, by assigning some rows to a fixed class, we can significantly reduce the number of symmetrical cases the SAT solver has to consider. We exploit this by precomputing a "partial solution". More specifically, we first compute a set $S = \{x_0, \ldots, x_k\}$ of pairwise incompatible rows. For all solutions of the SAT problem, each of these rows must be in a separate class. Thus, we can obtain an equisatisfiable formula by just assigning all elements of S to arbitrary, different classes. To this end, we replace the covering clauses of the rows in S by the clause $r_{x_0,0} \wedge r_{x_1,1} \wedge \cdots \wedge r_{x_k,k}$. Furthermore, for a row $x_i \in S$, we can remove all literals $r_{x_i,j}$ such that $j \neq i$ from the SAT formulation, and simplify the other constraints accordingly.

Moreover, we can also use the cardinality of S as a lower bound for the required number of classes.

If we represent the compatibility relation on the rows as a graph (s.t. there is an edge whenever two rows are compatible), the problem of finding such a set S corresponds to finding an independent set in the graph. While the problem of finding a maximum independent set is NP-hard, a heuristic is sufficient for our purposes, since a non-maximal set would just lead to a smaller reduction of symmetries, and to a smaller lower bound, but it would still lead to a correct solution.

We use the following simple sequential greedy heuristic to find a set of pairwise incompatible rows. We first create a list of all rows that is sorted in reverse order based on the number of incompatible rows for each row (or equivalently, the degree on the compatibility graph). The algorithm maintains a set S of pairwise independent states, where S is initially empty. We then iterate over the sorted list of rows. Whenever we encounter a row that is incompatible to all rows in S, we add this row to S. A similar approach was used by [3].

Excluding previously discovered partitions: Since our algorithm searches for multiple partitions for the same observation table, we add for each previously found partition a disjunction of the negated literals for this partition.

References

1. Abel, A., Reineke, J.: MeMin: SAT-based exact minimization of incompletely specified mealy machines. ICCAD '15, IEEE Press (2015)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Information and computation 75(2), 87–106 (1987)
3. Heule, M., Verwer, S.: Exact DFA identification using SAT solvers. In: Grammatical Inference: Theoretical Results and Applications, vol. 6339, pp. 66–79 (2010)