L^* and its Derivatives

Charles Averill Ben Kallus

Contents

1 Intuition

Given an arbitrary regular language L over the set of strings Σ^*, we would like to construct a DFA $(Q, \Sigma, \delta, q_0, F)$ that recognizes it. We will accomplish this by repeatedly incorporating feedback from a **Teacher** into an in-progress **Hypothesis DFA**. At each iteration, the Teacher will provide us with one of the following pieces of feedback:

1. The current hypothesis DFA recognizes L, or
2. The current hypothesis DFA fails to recognize L because it misclassifies the membership of string w (or, w is a **counterexample**).

The L^* algorithm and its derivatives learn the **minimal** DFA that recognizes L by maintaining an **observation table** (or similar data structure) of prefixes and **distinguishing suffixes** constructed from observed counterexamples. A prefix p maps to the DFA state q_p which can be reached by running the DFA on string p. But many unique strings will share common prefixes (and therefore end up at the same intermediate states) despite some being members of L and some not. A DFA must tell these two cases apart to encode L. Distinguishing suffixes are used for this purpose.

Each suffix s represents a membership query about a prefix ($ps \stackrel{?}{\in} L$). We can construct the **signature** of a prefix p by collecting the answers to the queries for p and each s. This core operation appeals to the Myhill-Nerode theorem:

Theorem 1. *For strings x, y, and z, let*

$$x \sim_L y \equiv (xz \in L \iff yz \in L).$$

*$\sim_L$ is an equivalence relation that partitions the set of all strings into **equivalence classes**, or sets of mutually $\sim_L$-equivalent strings. Language L is regular iff $\sim_L$ has finitely many equivalence classes. Furthermore, the minimal DFA accepting L has the same number of states as there are equivalence classes in L.*

Each equivalence class of $\sim_L$ therefore maps to exactly one state of the minimal DFA, and the signature of a prefix allows us to name which class it falls into. When $p \sim_L p'$, no suffix can ever tell them apart (so they must be in the same class, and therefore state), and when $p \not\sim_L p'$ due to some distinguishing suffix s, they must lie in distinct classes (and so must be in distinct states).

We can now construct a DFA from our observation table. Distinct signatures (or equivalence classes) map to distinct states. States are accepting when their prefix is in L. We can construct a transition function for state q_p and symbol a by computing the signature for prefix pa and transitioning to whichever state carries the matching signature ($\delta(q_p, a) \equiv q_{pa}$).

Whenever the Teacher returns a counterexample w (rather than approving of our hypothesis DFA), it indicates that the current set of distinguishing suffixes is insufficient (it failed to distinguish two prefixes that belong in distinct

equivalence classes). Thus, the current DFA merges two states that $\sim_L$ reports as distinct. To resolve this, we split w into a prefix and distinguishing suffix that refines the appropriate signatures, splitting the conflated class and thus requiring a new distinct DFA state. Because each counterexample reveals at least one new equivalence class, and Myhill-Nerode states that L only has finitely many such classes, we can repeatedly perform counterexample analysis with guaranteed termination. Upon termination, the hypothesis DFA has exactly the number of states as equivalence classes (and that of the minimal DFA accepting L), and the Teacher must then approve of the hypothesis.

There are several ways to implement observation tables, hypothesis DFAs, and the main learning loop. Three implementations are presented here:

- L^*: Dana Angluin's original implementation [Angluin(1987)]
- SepL^*: A modification of Angluin's implementation that simplifies the observation table structure and relies on different invariants [Managoli(2021)]
- Kearns-Vazirani: A derivative that encodes the observation table as a binary tree to minimize membership queries [Kearns and Vazirani(1994)]

2 L^*

Angluin's original L^* implementation organizes its observation table into two sets and a function:

- S: a nonempty, finite, prefix-closed set of prefixes
- E: a nonempty, finite, suffix-closed set of distinguishing suffixes, and
- T: a boolean function over $((S \cup S \cdot A) \cdot E)$, or, prefixes with or without a trailing symbol, followed by a distinguishing suffix. T is defined such that $T(u) = \mathit{true}$ iff $u \in L$.

The observation table is initially set to $S = E = \{\varepsilon\}$ and $T = \lambda x.\mathit{false}$. In this section, we will denote the set of symbols Σ as A.

2.1 Observation Tables

	s_1	s_2	...
p_1	*true*	*false*	...
p_2	*true*	*true*	...
p_3	*false*	*true*	...
...	...	...	...

Table 1: L^* observation table structure. Rows represent prefixes $p \in S$ (with or without trailing symbols), columns represent distinguishing suffixes in $s \in E$, and cell values are given by $T(se)$.

Table 1 shows the representation of an observation table as a two-dimensional array, where rows are indexed by an element of $(S \cup S \cdot A)$ and columns are indexed by an element of E. We can define $\text{row}(p)$ as $\lambda e.T(pe)$, essentially currying T. When assembling our DFA from the observation table, rows labeled by elements of S are candidates for states, and rows labeled by elements of $S \cdot A$ are used to construct the transition function. Cell values given by T define accepting states.

L^* maintains two key invariants throughout its learning loop: **closure** and **consistency**.

Definition 2.1. *A table (S, E, T) is **closed** iff*

$$\forall t \in S \cdot A,\ \exists s \in S, \text{row}(t) = \text{row}(s).$$

That is, for any row corresponding to a prefix with a trailing symbol, there exists a row with identical cells corresponding to an element of S.

Definition 2.2. *A table (S, E, T) is **consistent** iff*

$$\forall s_1, s_2 \in S, \text{row}(s_1) = \text{row}(s_2) \implies \forall a \in A, \text{row}(s_1 a) = \text{row}(s_2 a).$$

That is, for any two prefixes whose row values are identical, the rows of those prefixes followed by any symbol a are also identical.

We can construct a DFA $M(S, E, T)$ from any closed, consistent observation table as follows:

$$\begin{aligned} Q &\equiv \{\text{row}(s) \mid s \in S\}^1 \\ q_0 &\equiv \text{row}(\varepsilon) \\ F &\equiv \{\text{row}(s) \mid s \in S \wedge T(s) = \mathit{true}\} \\ \delta(\text{row}(s), a) &\equiv \text{row}(sa). \end{aligned}$$

We can also define a transition function for strings:

$$\delta^*(\text{row}(s_1), s_2) \equiv \begin{cases} \text{row}(s_1) & s_2 = \varepsilon \\ \delta(\delta^*(\text{row}(s_1), s_2'), a) & s_2 = s_2'a \end{cases}$$

Theorem 2.1. *$M(S, E, T)$ is a well-defined DFA.*

Proof. S is non-empty and prefix-closed, by definition. Thus, it necessarily includes ε, so q_0 is well-defined.

E is non-empty and suffix-closed, by definition, and so also includes ε. Consider any $s_1, s_2 \in S$ such that $\text{row}(s_1) = \text{row}(s_2)$. Trivially, $T(s_1) = T(s_1\varepsilon)$ and $T(s_2) = T(s_2\varepsilon)$. Because $\text{row}(s)$ evaluates to a sequence of $T(se)$ values, it must be true that $T(s_1) = T(s_2)$. Therefore, membership in F depends only on $\text{row}(s)$ and not on the choice of a representative s. Thus, it is not possible that for s_1, s_2 with $\text{row}(s_1) = \text{row}(s_2)$, $s_1 \in F \iff s_2 \in F$, so F is well-defined.

Consider any $s_1, s_2 \in S$ such that $\text{row}(s_1) = \text{row}(s_2)$. Since (S, E, T) is consistent, we know that $\forall a \in A$, $\text{row}(s_1a) = \text{row}(s_2a)$. Since the table is closed, we know that these rows are equal to $\text{row}(s)$ for some $s \in S$. Therefore the row $\text{row}(s_1a) = \text{row}(s_2a)$ depends only on $\text{row}(s_1)$ and a, not on the representative s_1, so δ assigns a unique value in Q to each (row, symbol) pair, so δ is well-defined. □

We would like to show that $M(S, E, T)$ is **minimal** and **consistent** with T for any closed and consistent observation table.

Definition 2.3. *A DFA $M(S, E, T)$ is **consistent** with T iff*

$$\forall u \in ((S \cup S \cdot A) \cdot E), \delta^*(q_0, u) \in F \iff T(u) = \mathit{true}.$$

That is, running the DFA on any string that labels a row provides the same answer as T.

Definition 2.4. *An acceptor $M(S, E, T)$ is **minimal** iff*

$$\forall M',\ M' \text{ consistent with } T \wedge M \ncong M' \implies |Q| < |Q'|.$$

That is, for any non-equivalent DFA M' that is consistent with T, the number of states in M is fewer than the number of states in M'.

[1] Here, although there may be many equivalent rows, constructing the **set** Q is deduplicating, and thus Q must contain no equivalent states.

Lemma 2.1. *Let (S, E, T) be closed and consistent. Consider the DFA $M(S, E, T)$.*

$$\forall s \in (S \cup S \cdot A), \delta^*(q_0, s) = \text{row}(s).$$

Proof. We will proceed by induction on the length of s.

Case 1: $|s| = 0$**.** It is necessary that $s = \varepsilon$. Trivially, $\delta^*(q_0, \varepsilon) = \text{row}(\varepsilon)$, by the definition of δ^* and because $q_0 = \varepsilon$.

Case 2: $|s| = k + 1$**.** Assume

$$\forall s' \in (S \cup S \cdot A),\ |s'| \leq k \implies \delta^*(q_0, s') = \text{row}(s')$$

by induction. We must show that $\delta^*(q_0, s'a) = \text{row}(s'a)$ for some a such that $s = s'a$. We know that $s' \in S$, because either

$$s'a \in S \cdot A \implies s' \in S, \text{ or}$$
$$s'a \in S \wedge S \text{ is prefix-closed} \implies s' \in S.$$

Then,

$$\begin{aligned} \delta^*(q_0, s'a) &= \delta(\delta^*(q_0, s'), a) && \text{by the definition of } \delta^* \\ &= \delta(\text{row}(s'), a) && \text{by the inductive hypothesis} \\ &= \text{row}(s'a) && \text{by the definition of } \delta. \end{aligned}$$

□

Lemma 2.2. *Let (S, E, T) be closed and consistent. $M(S, E, T)$ is consistent with the finite function T.*

Proof. To prove that M is consistent w.r.t. T, we must show that

$$\forall s \in (S \cup S \cdot A), e \in E, \delta^*(q_0, se) \in F \iff T(se) = \mathit{true}.$$

We will proceed by induction on the length of e.

Case 1: $|e| = 0$**.** By Lemma 2.1, $\delta^*(q_0, s) = \text{row}(s)$.

We can show $\text{row}(s) \in F \iff T(s) = \mathit{true}$ by

Case 1.1: $s \in S$**.** By the definition of F: $\text{row}(s) \in F \iff T(s) = \mathit{true}$.

Case 1.2: $s \in S \cdot A$**.** Because the table is closed, we know that $\exists s' \in S, \text{row}(s) = \text{row}(s')$. Furthermore, $\text{row}(s') \in F \iff T(s') = \mathit{true}$, so it must be that $T(s) = \mathit{true}$.

Case 2: $|e| = k + 1$**.** Assume

$$\forall s \in (S \cup S \cdot A), e' \in E, |e'| \leq k \implies \delta^*(q_0, se) \in F \iff T(se') = \mathit{true}$$

by induction. Because E is suffix-closed, we know that $e = ae'$ for some $a \in A, e' \in E$. Because the table is closed, it must be that $\exists s', \text{row}(s) = \text{row}(s')$.

$$\begin{aligned}
\delta^*(q_0, se) &= \delta^*(q_0, sae') \\
&= \delta^*(\delta^*(q_0, s), ae') && \text{by definition of } \delta^* \\
&= \delta^*(\text{row}(s), ae') && \text{by Lemma 2.1} \\
&= \delta^*(\text{row}(s'), ae') && \text{since } \text{row}(s) = \text{row}(s') \\
&= \delta^*(\delta(\text{row}(s'), a), e') \\
&= \delta^*(\text{row}(s'a), e') && \text{by the definition of } \delta \\
&= \delta^*(\delta^*(q_0, s'a), e') && \text{by Lemma 2.1} \\
&= \delta^*(q_0, s'ae').
\end{aligned}$$

By the inductive hypothesis, we know that $\delta^*(q_0, s'ae') \in F \iff T(s'ae') = \textit{true}$. Because $\text{row}(s) = \text{row}(s')$, we know that $T(s'ae') = T(sae') = T(se)$. Therefore, $\delta^*(q_0, se) \in F \iff T(se) = \textit{true}$.

□

Lemma 2.3. *Let (S, E, T) be closed and consistent. Consider the DFA $M(S, E, T)$ with $|Q| = n$. If another DFA M' is consistent with T and $|Q'| \leq n$, then $M \cong M'$.*

Proof. We will proceed by showing an isomorphism between M and M'. That is, we will produce a map $\varphi : Q \to Q'$ that

1. Is injective: $\forall q_1, q_2 \in Q, q_1 \neq q_2 \implies \varphi(q_1) \neq \varphi(q_2)$ and surjective. Since $|Q'| \leq n = |Q|$, injectivity forces $|Q'| = n$, so an injective φ is automatically a bijection.
2. Preserves the start state: $\varphi(q_0) = q_0'$,
3. Commutes with transitions: $\forall \text{row}(s) \in Q, a \in A, \varphi(\delta(\text{row}(s), a)) = \delta'(\varphi(\text{row}(s)), a)$, and
4. Preserves acceptance: $\forall q \in Q, q \in F \iff \varphi(q) \in F'$.

Let $\varphi(\text{row}(s)) = \delta'^*(q_0', s)$.

φ is injective. Let $\text{row}(s_1), \text{row}(s_2) \in Q$ with $s_1, s_2 \in S$ and $\text{row}(s_1) \neq \text{row}(s_2)$. By the definition of $\text{row}(s)$, $\exists e \in E, T(s_1e) \neq T(s_2e)$. Because M' is consistent with T, $\delta'^*(q_0', s_ie) \in F' \iff T(s_ie) = \textit{true}$. We can show

$$\begin{aligned}
\delta'^*(q_0', s_ie) &= \delta'^*(\delta'^*(q_0', s_i), e) && \text{by the transitivity of } \delta'^* \\
&= \delta'^*(\varphi(\text{row}(s_i), e).
\end{aligned}$$

Therefore,

$$\delta'^*(\varphi(\text{row}(s_i)), e) \in F' \iff T(s_ie) = \textit{true}.$$

Because $T(s_1e) \neq T(s_2e)$, exactly one of $\delta'^*(\varphi(\text{row}(s_1)), e))$, $\delta'^*(\varphi(\text{row}(s_2)), e))$ exists in F'. If it were that $\varphi(\text{row}(s_1)) = \varphi(\text{row}(s_2))$, then feeding both the same suffix e would land in the same state, and hence agree on membership in F'. Thus, we have a contradiction, and φ is injective (and, by the restraint on $|Q'|$, surjective).

φ **preserves** q_0**.**

$$\begin{aligned}\varphi(q_0) &= \varphi(\text{row}(\varepsilon)) \\ &= \delta'^*(q'_0, \varepsilon) \\ &= q'_0\end{aligned}$$

φ **commutes with transitions.** Let $s' \in S$ such that $\text{row}(sa) = \text{row}(s')$. Then,

$$\begin{aligned}\varphi(\delta(\text{row}(s), a)) &= \varphi(\text{row}(sa)) \\ &= \varphi(\text{row}(s')) \\ &= \delta'^*(q'_0, s').\end{aligned}$$

Also,

$$\begin{aligned}\delta'(\varphi(\text{row}(s)), a) &= \delta'(\delta'^*(q'_0, s), a) \\ &= \delta'(q'_0, sa).\end{aligned}$$

$\delta'^*(q'_0, s')$ and $\delta'(q'_0, sa)$ have identical row values ($\text{row}(s')$ and $\text{row}(sa)$). Since φ is a bijection, rows identify unique states of M'. Hence the two are the same state.

φ **preserves acceptance.** Let $\text{row}(s) \in Q$ with $s \in S$. By the definition of F, and because $\varepsilon \in E$ so that $T(s) = T(s\varepsilon)$ is an entry of $\text{row}(s)$,

$$\text{row}(s) \in F \iff T(s) = \textit{true}.$$

Since M' is consistent with T, taking the suffix $\varepsilon \in E$ gives

$$\varphi(\text{row}(s)) = \delta'^*(q'_0, s) \in F' \iff T(s\varepsilon) = T(s) = \textit{true}.$$

Chaining these biconditionals through $T(s) = \textit{true}$,

$$\text{row}(s) \in F \iff T(s) = \textit{true} \iff \varphi(\text{row}(s)) \in F'.$$

Hence $\text{row}(s) \in F \iff \varphi(\text{row}(s)) \in F'$, so φ preserves acceptance.

□

Theorem 2.2. *$M(S, E, T)$ is minimal and consistent with T if (S, E, T) is closed and consistent.*

Proof. M is consistent by Lemma 2.2. By Lemma 2.3, any other DFA consistent with T is either isomorphic to or larger than M. Thus, M is the smallest DFA consistent with T. □

2.2 L^* Learner

Algorithm 1 The L^* learning loop

$S \leftarrow E \leftarrow \{\varepsilon\}$
$T \leftarrow$ `fold_left` $(\lambda f, a.f[a \coloneqq a \in L])$ $(\lambda x.false)$ $(\{\varepsilon\} \cup A)$
repeat
  while (S, E, T) not closed or not consistent **do**
    if (S, E, T) not consistent **then**
      Find $s_1, s_2 \in S,\ a \in A,\ e \in E$ such that
        $\text{row}(s_1) = \text{row}(s_2), T(s_1ae) \neq T(s_2ae)$
      Add ae to E
      Extend T using membership queries
    if (S, E, T) not closed **then**
      Find $s' \in S,\ a \in A$ such that
        $\forall s \in S,\ \text{row}(s'a) \neq \text{row}(s)$
      Add $s'a$ to S
      Extend T using membership queries
  $M \leftarrow M(S, E, T)$
  match Query Teacher **with**
    M recognizes $L \Rightarrow$
      break
    w is a counterexample $\Rightarrow$
      Add w and all of its prefixes to S
      Extend T using membership queries
  end match
return M

Algorithm 1 shows the L^* learner loop. Each iteration, the loop closes and makes consistent the observation table (S, E, T). Once the table is closed and consistent, the Teacher is queried with $M(S, E, T)$. If the teacher replies that M is correct, the algorithm terminates. Otherwise, the learner incorporates the counterexample w into S and T and then loops to close and make consistent the observation table.

2.3 Correctness

Remark 2.1. *L^* is trivially correct if it terminates, because termination occurs only when the (trusted) teacher states that M is correct.*

Theorem 2.3. *L^* terminates.*

Proof. L is regular. By Myhill-Nerode, there are therefore a finite number of equivalence classes given by $\sim_L$. Furthermore, there exists a minimal DFA recognizing L with $|Q| = n$.

We first observe that the number of distinct rows in any observation table produced by a partial execution of L^* is at most n. Each $\text{row}(s)$ is determined by

the membership of $se \in L$ over $e \in E$, so any two prefixes in the same $\sim_L$ class have equal rows. Thus, distinct rows correspond to distinct equivalence classes of $\sim_L$, of which there are exactly n. Therefore $|Q| = |\{\text{row}(s) \mid s \in S\}| \leq n$ at every iteration.

Next, we can show that each table modification strictly increases the number of distinct rows. Upon closing a table, we add some $s'a$ to S where $\forall s \in S$, $\text{row}(s'a) \neq \text{row}(s)$. Upon making a table consistent, we add a suffix ae to E such that $\text{row}(s_1a) \neq \text{row}(s_2a)$ where previously $\text{row}(s_1) = \text{row}(s_2)$. Finally, receiving a counterexample is proof that $|Q| < n$. Incorporating counterexample feedback must expose at least one distinct row, otherwise M would be unchanged and would still misclassify w, yet the table would, by Lemma 2.3, yield the minimal DFA consistent with the now-expanded T, resulting in a contradiction.

Because the number of distinct rows both strictly increases and is always bounded by n, L^* must terminate. □

3 SepL^*

The L^* algorithm presented in Section 2 maintains two invariants: **closure** and **consistency**. While correct, the dependence on these invariants has a negative impact on both memory usage and performance when they must be repaired after incorporating counterexample feedback. When consistency is broken, the algorithm searches over pairs of prefixes with equivalent rows and symbols in Σ. This issue is made worse by the allowance of L^* for duplicate rows in the observation table, which wastes space and increases consistency search time.

SepL^* presents an alternative invariant strategy to minimize these inconveniences. Rather than maintaining a set of rows that are de-duplicated in each loop iteration, SepL^* maintains a set of access strings and a set of distinguishing suffixes $Q, T \subseteq \Sigma^*$ with an invariant that the elements of Q are **pairwise T-distinguishable**. This property, called **separability**, replaces the consistency invariant: where L^* repairs inconsistencies by adding suffixes to E, SepL^* never admits strings into its state set unless they are already distinguished by a suffix. Note that Q and T are analogous to the roles of S and E in L^*, but are simply sets of strings, rather than indices into a table.

3.1 T-equivalence

The T-equivalence relation is a refinement of the $\sim_L$ relation, relative to the set T rather than L as in Myhill-Nerode.

Definition 3.1. *Strings u, v are **T-equivalent** (denoted $u \equiv_T v$) iff*

$$\forall t \in T,\ ut \in L \iff vt \in L.$$

*Otherwise, they are **T-distinguishable** ($u \not\equiv_T v$.)*

Remark 3.1. *$\equiv_T$ is an equivalence relation.*

Lemma 3.1. *If $T_1 \subseteq T_2$, then $\equiv_{T_2}$ refines $\equiv_{T_1}$. That is,*

$$\forall u, v, u \equiv_{T_2} v \implies u \equiv_{T_1} v.$$

Proof. Let $t \in T_1$. By the subset assumption, $t \in T_2$, so $ut \in L \iff vt \in L$, and thus $u \equiv_{T_1} v$. □

Remark 3.2. *The relation $\equiv_{\Sigma^*}$ (equivalent to $\sim_L$) refines every $\equiv_T$. Therefore, any two T-distinguishable strings for some finite T are Myhill-Nerode distinct.*

Lemma 3.2. *For finite T, $u \equiv_T v$ is decidable.*

Proof. One can iterate over each $t \in T$ and check if $ut \in L \iff vt \in L$ holds by making membership queries to the Teacher. □

3.2 Invariants

SepL^*'s two structural properties are **closure** (analogous to Definition 2.1) and **separability**.

Definition 3.2. *Q is **closed** with respect to T iff*

$$\forall q \in Q, a \in \Sigma, \exists q' \in Q, qa \equiv_T q'.$$

That is, every one-symbol extension of an access string has a T-equivalent representative in Q.

Definition 3.3. *Q is **separable** with respect to T iff*

$$\forall u, v \in Q, u \neq v \implies u \not\equiv_T v.$$

That is, distinct states in Q must be T-distinguishable.

Separability removes the need for de-duplication of the observation table, saving on both memory and time complexity. Closure guarantees that transitions stay inside Q (and will thus soon be used to define the transition function).

Remark 3.3. *For finite Q, T, closure and separability are decidable.*

3.3 DFA Construction

From a closed and separable observation table (Q, T) with $\varepsilon \in Q$, we can construct a DFA whose states are exactly the set Q.

The transition function is given by closure:

$$\delta(q, a) \equiv \textit{the unique } q' \in Q \textit{ such that } qa \equiv_T q'.$$

Uniqueness is given by separability (there can be no two states in Q that are T-equivalent to qa).

Constructing a DFA from the observation table is given by

$$\begin{aligned} Q &\equiv Q \\ q_0 &\equiv \varepsilon \\ F &\equiv \{q \in Q \mid q \in L\}^2 \\ \delta &\equiv \delta \qquad\qquad \text{as given above.} \end{aligned}$$

[2] Access to the Teacher after constructing the hypothesis DFA may not be guaranteed in some contexts, so this set may be constructed immediately by mapping a membership query over Q.

3.4 Counterexample Analysis

SepL^* utilizes a similar approach to L^* for incorporating counterexample feedback. At each iteration, a counterexample is added to the observation table, and then separability and closure are maintained by adding additional strings as necessary.

Theorem 3.1. *Let (Q, T) be closed and separable, and let w be a counterexample for $M(Q, T)$. There exists an index i that partitions w and a prefix $p = \delta^*(\varepsilon, w[:i])$ such that $(Q \cup \{p \cdot w[i]\})$ is separable with respect to $(T \cup \{w[i{+}1:]\})$ and $p \cdot w[i] \notin Q$.*

Proof. For $j \in [0, m]$ where $m = |w|$, let

$$\pi(j) \equiv \delta^*(\varepsilon, w[:j]) \in Q$$

be the state $M(Q, T)$ reaches on the length-j prefix of w, so that $p = \pi(i)$ for the index i we will fix next. A state $\pi(j)$ is considered **correct** iff

$$\pi(j)w[j:] \in L \iff w \in L.$$

Correctness is decidable with membership queries to the Teacher.

$\exists i, \mathit{correct}(i) \wedge \neg \mathit{correct}(i+1)$**.** $\pi(0)$ is trivially correct ($w \in L \iff w \in L$ is tautological). $\pi(m)$ is trivially incorrect ($M(w) \in F \not\iff w \in L$ because w is a counterexample). Therefore, there must exist an $i \in [0, m)$ such that it partitions w into a correct prefix and incorrect suffix. Because correctness is decidable, i can be computed by traversing w and computing correctness of its partitionings. Select $p \equiv \pi(i)$.

$w[i+1:]$ **distinguishes** $p \cdot w[i]$ **from** $\pi(i+1)$**.**

By the definition of δ, $\pi(i+1) = \delta(\pi(i), w[i])$ is the representative in Q with $\pi(i+1) \equiv_T \pi(i) \cdot w[i] = p \cdot w[i]$. Correctness of i gives

$$p \cdot w[i] \cdot w[i+1:] = \pi(i) \cdot w[i:] \in L \iff w \in L,$$

using $w[i] \cdot w[i+1:] = w[i:]$. Incorrectness of $i+1$ gives

$$\pi(i+1) \cdot w[i+1:] \in L \not\iff w \in L.$$

To restate the goal, $p \cdot w[i] \cdot w[i+1:] \in L \not\iff \pi(i+1) \cdot w[i+1:] \in L$. Thus the single suffix $w[i+1:]$ witnesses $p \cdot w[i] \not\equiv_{T'} \pi(i+1)$, with T' defined as $(T \cup \{w[i+1:]\})$.

$p \cdot w[i] \notin Q$**.** Suppose $p \cdot w[i] \in Q$. Then $p \cdot w[i]$ and $\pi(i+1)$ both lie in Q, and $\pi(i+1) \equiv_T p \cdot w[i]$ by construction. This forces $p \cdot w[i] = \pi(i+1)$ by the separability of Q. But then, $p \cdot w[i] \cdot w[i+1:] = \pi(i+1) \cdot w[i+1:]$, but these strings disagree on membership. Thus, we have a contradiction.

$(Q \cup \{p \cdot w[i]\})$ **is separable with respect to** $(T \cup \{w[i+1{:}]\})$**.**

Let $T' = (T \cup \{w[i+1{:}]\})$ and take distinct $u, v \in (Q \cup \{p \cdot w[i]\})$. If both lie in Q, separability of Q gives a suffix in $T \subseteq T'$ distinguishing them, and refinement preserves this under T'. Otherwise one is the new state, such as $u = p \cdot w[i]$ and $v \in Q$. Suppose $p \cdot w[i] \equiv_{T'} v$. By refinement, $p \cdot w[i] \equiv_T v$, and since $\pi(i+1) \equiv_T p \cdot w[i]$, transitivity gives $\pi(i+1) \equiv_T v$ with both in Q, so separability forces $v = \pi(i+1)$. But $w[i+1{:}] \in T'$ distinguishes $p \cdot w[i]$ from $\pi(i+1) = v$, a contradiction.

□

After incorporating counterexample feedback, we only gain that the resulting table is separable, but not necessarily closed. This is because the new addition to Q, $p \cdot w[i]$ may have single-symbol extensions with no representative. We can close Q while maintaining separability by adding these representatives. Following is a lemma regarding the selection of such a representative, and a theorem stating that we can close Q with these representatives.

Lemma 3.3. *Let (Q, T) be separable. For all $q \in Q, a \in \Sigma$, there exists a set Q' separable with T such that $Q = Q' \vee Q' = (Q \cup \{qa\})$, and in either case, $\exists q' \in Q', qa \equiv_T q'$.*

Proof. Decide whether $\exists r \in Q, qa \equiv_T r$. If so, take $Q' \equiv Q$ (trivially satisfying the final condition in the lemma statement).

If not, add qa to Q. Separability is preserved because qa is T-distinguishable from every $r \in Q$ by the fact that we have already decided that such a string does not exist. Then, $q' \equiv qa$. □

Theorem 3.2. *Let (Q, T) be separable. Finitely many applications of Lemma 3.3 produce a set Q' with $Q \subseteq Q'$ that is closed and separable with respect to T.*

Proof. Each closure step that augments Q adds exactly one element which is T-distinguishable from all existing elements, so $|Q|$ strictly increases while preserving separability. There is an injection from a separable set Q to the set of $\equiv_T$-equivalence classes (each element in Q maps to a single $\equiv_T$-equivalence class). Define the membership vector for a string u to be

$$\langle ut_1 \stackrel{?}{\in} L, ut_2 \stackrel{?}{\in} L, \ldots, ut_k \stackrel{?}{\in} L \rangle \in \{true, false\}^{|T|}.$$

Two strings u, v have the same membership vector iff they agree on every $t \in T$, *i.e.*, when $u \equiv_T v$. Therefore, the membership vector is a label for a string's $\equiv_T$-equivalence class.

Because a separable Q has all of its elements in distinct $\equiv_T$-classes, each member has a unique membership vector, giving an injection from Q onto $\{true, false\}^{|T|}$, a set with $2^{|T|}$ elements. So $|Q| \leq 2^{|T|}$ at each application of the closure step, and since each step strictly increases $|Q|$ (or terminates with $Q' = Q$), only finitely many steps occur before Q becomes closed. □

Thus, each time the Teacher provides the Learner with a counterexample, we can always incorporate its feedback while maintaining closure and separability.

3.5 Hypothesis Correctness

We can now show that $M(Q,T)$ for a closed and separable table must never be larger than the minimal DFA recognizing L, and that if the teacher fails to provide a counterexample, M must be minimal.

Lemma 3.4. *Let (Q,T) be closed and separable. Consider the DFA $M(Q,T)$. Let D be any DFA that recognizes L. $|Q| \leq n \leq |Q_D|$, where n is the number of states in the minimal DFA.*

Proof. Map each string $q \in Q$ onto the state $\delta_D^*(q_{0,d}, q)$ of D.

This map is injective. Suppose $u, v \in Q$ reach the same state of D. Then for every suffix $t \in T$, $D(ut) \in F_D \iff D(vt) \in F_D$. Because D recognizes L, this gives $ut \in L \iff vt \in L$ for all t. Thus $u \equiv_T v$, and by separability, $u = v$.

$|Q| \leq n$. An injection from Q into Q_D gives $|Q| \leq |Q_D|$. Applying this knowledge when D is the minimal DFA shows that $|Q| \leq n$.

□

Theorem 3.3. *Let (Q,T) be closed and separable. Consider the DFA $M(Q,T)$. If the Teacher reports no counterexample, then $M(Q,T)$ is a minimal DFA encoding L.*

Proof. M encodes L trivially, by the nature of the Teacher reporting no counterexample. By Lemma 3.4, M is minimal. □

3.6 SepL^* Learner

Algorithm 2 The SepL^* learning loop

```
Q ← {ε}
T ← ∅
repeat
    while (Q, T) not closed do
        Find q ∈ Q, a ∈ A such that ∀q' ∈ Q, qa ≢_T q'
        Add qa to Q
    M ← M(Q, T)
    match Query Teacher with
        M recognizes L ⇒
            break
        w is a counterexample ⇒
            Find partitioning i of w in M
            Q ← (Q ∪ {π(i) · w[i]})
            T ← (T ∪ {w[i + 1:]})
    end match
return M
```

Algorithm 2 shows the SepL^* learner loop. Each outer iteration restores closure by Theorem 3.2, then queries the Teacher with $M(Q, T)$. If the teacher replies that M is correct, the algorithm terminates. Otherwise, a counterexample w is analyzed to find a useful partitioning by Theorem 3.1. Separability is never violated, and so never needs a repair loop like closure.

Remark 3.4. SepL^* *is trivially correct if it terminates, because termination occurs only when the (trusted) Teacher states that* M *is correct.*

Theorem 3.4. SepL^* *terminates.*

Proof. L is regular, so by Myhill-Nerode there is a minimal DFA recognizing L with n states.

The inner closure loop terminates by Theorem 3.2. By Lemma 3.4, $|Q| \leq n$ at each iteration. By Theorem 3.1, each outer iteration adds a unique string to Q derived from w that is T-distinguishable from all existing strings in Q. Adding suffixes to T only refines $\equiv_T$, never making (Q, T) inseparable and thus never making $|Q|$ decrease. Because $|Q|$ strictly increases and is bounded by n, the outer loop terminates. □

4 Kearns-Vazirani

Both L^* and SepL^* store their knowledge in flat structures such as a table of rows or a pair of string sets that encode no relationships between distinguishing suffixes. Classifying a string then requires comparing it against every distinguishing suffix, which is costly. The Kearns-Vazirani algorithm (henceforth referred to as KV) replaces these flat structures with a **discrimination tree**, a binary tree that organizes distinguishing suffixes hierarchically so that a string is classified by a sequence of membership queries along a single root-to-leaf path, rather than against all suffixes. A string reaches its state after one query per node on the path, and the tree's depth is at most the number of states. Thus, traversing the tree during learning generally requires fewer membership queries (as few as logarithmically many) than in the previously-described algorithms (linear).

4.1 Discrimination Trees

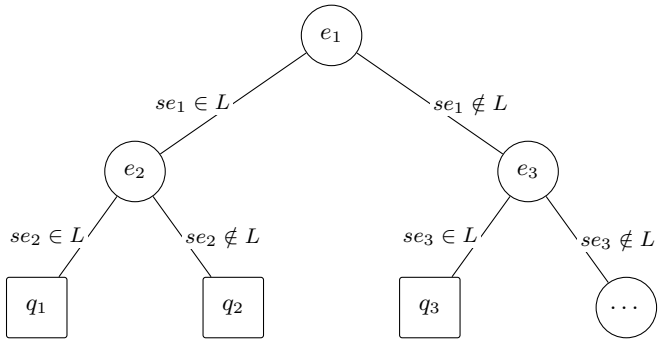

Figure 1: Discrimination tree structure. Internal nodes hold discriminators $e \in \text{discriminators}(t)$ and leaves hold access strings $q \in \text{leaves}(t)$. To classify a string s, sift it from the root: at an internal node with discriminator e, descend the left branch when $se \in L$ and the right branch when $se \notin L$. The leaf reached names the state of s.

Figure 1 shows the structure of a discrimination tree. Internal nodes carry a **discriminator** (distinguishing suffix), and leaves carry an **access string** (state representative). A string is classified by **sifting** it down the tree. At each internal node with discriminator e, we ask $ue \stackrel{?}{\in} L$ and descend left if it holds, right otherwise. The access string at the leaf reached is the corresponding DFA state of running u from q_0.

Definition 4.1. *The **sift** of a string u through a tree t is given by*

$$\mathrm{sift}(t,u) \equiv \begin{cases} q & t = \mathrm{Leaf}(q) \\ \mathrm{sift}(l,u) & t = \mathrm{Node}(e,l,r) \wedge ue \in L \\ \mathrm{sift}(r,u) & t = \mathrm{Node}(e,l,r) \wedge ue \notin L \end{cases}$$

We write $\mathrm{leaves}(t)$ for the list of access strings at the leaves of t, and $\mathrm{discriminators}(t)$ for the list of discriminators at its internal nodes.

Remark 4.1. *Sifting always terminates on a leaf.*

4.2 Invariants

KV maintains three properties relating the tree's leaves to its discriminators. The first two, **consistency** and **separability** are analogues of consistency in L^* and separability in $\mathrm{Sep}L^*$. The third, **well-formedness**, is necessary and sufficient for consistency if there are no duplicate leaves, and is the invariant maintained by the algorithm (the others are derived as needed).

Definition 4.2. *A tree t is **consistent** iff*

$$\forall q \in \mathrm{leaves}(t),\ \mathrm{sift}(t,q) = q.$$

That is, if all access strings shift to their own leaf.

Definition 4.3. *A tree t is **separated** iff*

$$\forall u, v \in \mathrm{leaves}(t), u \neq v \implies \exists e \in \mathrm{discriminators}(t),\ ue \in L \iff ve \in L.$$

That is, if any two leaves are told apart by some discriminator in the tree.

Definition 4.4. *A tree t is **well-formed** iff*

$$\mathrm{wf}(t) \equiv \varepsilon \in \mathrm{leaves}(t) \wedge \begin{cases} \mathit{true} & t = \mathrm{Leaf}(q) \\ (\forall q \in \mathrm{leaves}(l),\ qe \in L) \wedge & \\ \quad (\forall q \in \mathrm{leaves}(r),\ qe \notin L) \wedge & \\ \quad \mathrm{wf}(l) \wedge \mathrm{wf}(r) & t = \mathrm{Node}(e,l,r). \end{cases}$$

That is, if at every internal node, the discriminator bisects the behavior of the two subtrees, and the subtrees are well-formed.

Lemma 4.1. *If t is well-formed, then t has no duplicate leaves, is consistent, and is separated.*

Proof. **t contains no duplicate leaves.** By induction on t. At any $\mathrm{Node}(e,l,r)$, well-formedness makes every leaf of l satisfy $qe \in L$ and every leaf of r satisfy $qe \notin L$. Thus, the two leaf sets must be disjoint. l and r are each duplicate-free by the inductive hypothesis, so $\mathrm{leaves}(t)$ must contain no duplicates.

t **is consistent.** By induction on t. At a node, consider $q \in \text{leaves}(t)$. If q is a left leaf, then $qe \in L$ by $\text{wf}(l)$, so sifting descends left and $\text{sift}(l, q) = q$ by induction. An analogous argument holds if q is a right leaf.

t **is separated.** By induction on t. Consider two distinct leaves of $\text{Node}(e, l, r)$. If they lie in different subtrees, then the node's own discriminator e separates them (since one satisfies $qe \in L$ and the other does not). If they lie in the same subtree, the inductive hypothesis supplies a discriminator within that subtree, which is also a discriminator of t.

□

Lemma 4.2. *If t has no duplicate leaves and is consistent, then t is well-formed.*

Proof. By induction on t. Consider any left leaf $q \in \text{leaves}(l)$ for a node $\text{Node}(e, l, r)$. Consistency gives $\text{sift}(t, q) = q$. Were $qe \notin L$, sifting would descend right, landing on a leaf of $q' = q \in \text{leaves}(r)$, but then q would appear as a duplicate leaf, a contradiction. Hence, $qe \in L$, and symmetrically every right leaf satisfies $qe \notin L$. By the inductive hypothesis, l and r are well-formed. □

4.3 DFA Construction

From a tree t we can construct a DFA M whose states are the set $\text{leaves}(t)$:

$$\begin{aligned} Q &\equiv \text{leaves}(t) \\ q_0 &\equiv \text{sift}(t, \varepsilon) \\ F &\equiv \{q \in Q \mid q \in L\} \\ \delta(q, a) &\equiv \text{sift}(t, qa). \end{aligned}$$

Remark 4.1 guarantees that q_0 and every transition target are leaves in Q, so $M(t)$ is well-defined for any tree.

4.4 Counterexample Analysis

Counterexamples are incorporated by **splitting** a leaf. A counterexample occurs when a string reaches a leaf that should have discriminated it into its correct equivalence class, but did not. Thus, we can replace the offending leaf with a new internal node that discriminates between the old string and the newly-discovered counterexample. This analysis will add exactly one state.

Definition 4.5.

$$\text{split}(t, q_t, e, q') \equiv \begin{cases} \text{Node}(e', \text{split}(lt, q_t, e, q'), \text{split}(rt, q_t, e, q')) & t = \text{Node}(e', lt, rt) \\ \text{Leaf}(q) & t = \text{Leaf}(q) \wedge q \neq q_t \\ \text{Node}(e, \text{Leaf}(q_t), \text{Leaf}(q')) & t = \text{Leaf}(q_t) \wedge q_t e \in L \\ \text{Node}(e, \text{Leaf}(q'), \text{Leaf}(q_t)) & t = \text{Leaf}(q_t) \wedge q_t e \notin L \end{cases}$$

The tree $\text{split}(t, q_t, e, q')$ *is obtained by replacing* $\text{Leaf}(q_t)$ *with a node on* e *whose two children are* $\text{Leaf}(q_t)$ *and* $\text{Leaf}(q')$*, oriented for well-formedness.*

Theorem 4.1. *Consider a well-formed t with $\varepsilon \in \text{leaves}(t)$. Let w be a counterexample for $M(t)$.*

$$\exists\ q_t \in \text{leaves}(t), e \in \Sigma^*, q' \notin \text{leaves}(t),$$
$$q_t e \notin L \iff q' e \in L \wedge t' \equiv \text{split}(t, q_t, e, q') \text{ is well-formed.}$$

Proof. For $j \in [0, m]$, where $m = |w|$, let

$$\pi(j) \equiv \text{sift}(t, w[:j])$$

be the access string $M(t)$ reaches on the length-j prefix of w. A string $\pi(j)$ is considered **correct** iff

$$\pi(j)w[j:] \in L \iff w \in L,$$

exactly as defined in Section 2.

By Lemma 4.1, t is consistent and free of duplicate leaves. $\pi(0)$ is correct ($\pi(0) = \text{sift}(t, \varepsilon) = \varepsilon w = w$). $\pi(m)$ is incorrect ($\pi(m)w[m:] = \pi(m) \in L \not\iff w \in L$, because w is a counterexample). Thus, there must exist a computable $i \in [0, m)$ such that it partitions w into a correct prefix and incorrect suffix.

Set $q' \equiv \pi(i)w[i]$, $e \equiv w[i+1:]$, $q_t \equiv \text{sift}(t, q')$. One more step of M advances from $\pi(i)$ to q_t ($\pi(i+1) = \text{sift}(t, w[:i+1]) = \text{sift}(t, q') = q_t$).

e separates q_t from q'. Correctness of $\pi(i)$ gives

$$q'e = \pi(i)w[i:] \in L \iff w \in L.$$

Incorrectness of $\pi(i+1)$ gives

$$q_t e = \pi(i+1)w[i+1:] \in L \not\iff w \in L.$$

Hence, $q_t e \in L \not\iff q' e \in L$.

$q' \notin \text{leaves}(t)$. If $q' \in \text{leaves}(t)$, then by consistency $\text{sift}(t, q') = q'$, so $q' = q_t$. The two would then not be separated by e, a contradiction.

t' is well-formed. $\varepsilon \in \text{leaves}(t')$ because $\varepsilon \in \text{leaves}(t)$, and splitting only augments the leaf set by definition.

q_t is the leaf that q' reaches, and the split resulting in t' orients the new node by $q_t e \in L \not\iff q' e \in L$. Replacing the split leaf with the correctly-oriented node preserves the bisection condition at every ancestor node (the traversal of q' to q_t certifies that q' agrees with every discriminator on the path to q_t, so it must join the correct subtree everywhere above).

□

4.5 Hypothesis Correctness

Remark 4.2. *Splitting a leaf always increases the total leaf count by 1.*

As with SepL^*, the hypothesis DFA is never larger than the minimal DFA (with $|Q_D| = n$), because separation injects the leaves into the states of any DFA recognizing L.

Lemma 4.3. *Consider a well-formed tree t. Let D be any DFA recognizing L.*

$$|\text{leaves}(t)| \leq |Q_D|.$$

Proof. Map each $q \in \text{leaves}(t)$ onto the state $\delta_D^*(q_{0,D}, q)$ that D reaches on q. If two leaves u, v reach the same state of D, then $ut \in L \iff vt \in L$ for all t. By separation (given by Lemma 4.1), this would be contradictory unless $u = v$. Thus, this map is injective, and $|\text{leaves}(t)| \leq |Q_D|$. Specializing D to the minimal DFA bounds $|\text{leaves}(t)|$ by n. □

Theorem 4.2. *Consider a well-formed tree t and the DFA $M(t)$. If the Teacher reports no counterexample, then $M(t)$ is a minimal DFA encoding L.*

Proof. M encodes L trivially, by the nature of the Teacher reporting no counterexample. By Lemma 4.3, M is minimal because it contains exactly $|\text{leaves}(t)|$ states by construction. □

4.6 Kearns-Vazirani Learner

Algorithm 3 The Kearns-Vazirani learning loop

```
t ← Leaf(ε)
repeat
    M ← M(t)
    match Query Teacher with
        M recognizes L ⇒
            break
        w is a counterexample ⇒
            Find partitioning i of w in M
            q_new ← π(i)w[i]
            q_target ← sift(t, q_new)
            e ← w[i + 1:]
            t ← split(t, q_target, e, q_new)
    end match
```

Algorithm 3 shows the Kearns-Vazirani learner loop. The tree is seeded with the leaf ε, which is trivially well-formed. Each iteration queries the teacher with $M(t)$. If the teacher replies that M is correct, the algorithm terminates. Otherwise, a counterexample w is analyzed to find a useful partitioning by Theorem 4.1, which splits a leaf, adding a new state and discriminator.

Remark 4.3. *Kearns-Vazirani is trivially correct if it terminates, because termination occurs only when the (trusted) Teacher states that M is correct.*

Theorem 4.3. *Kearns-Vazirani terminates.*

Proof. L is regular, so by Myhill-Nerode there is a minimal DFA recognizing L with n states.

By Lemma 4.3, $|Q| \leq n$ at each iteration. By Remark 4.2, each iteration adds a unique string to Q derived from w. Splitting never removes a leaf, so $|Q|$ never decreases. Because $|Q|$ strictly increases and is bounded by n, the loop terminates. □

References

[Angluin(1987)] Dana Angluin. 1987. Learning regular sets from queries and counterexamples. **Inf. Comput.** 75, 2 (Nov. 1987), 87–106. doi:10.1016/0890-5401(87)90052-6

[Kearns and Vazirani(1994)] Michael J. Kearns and Umesh Virkumar. Vazirani. 1994. **An introduction to computational learning theory**. MIT Press, Cambridge, Mass. https://doi.org/10.7551/mitpress/3897.001.0001

[Managoli(2021)] Malhar Managoli. 2021. The L* Algorithm. https://www.tifr.res.in/~shibashis.guha/courses/diwali2021/L-starMalharManagoli.pdf