Synthesis and Verification of Register Automata over Data Sequences

Chih-Duo Hong[1], Hongjian Jiang[2], Anthony W. Lin[3,4], Oliver Markgraf[2], Julian Parsert[2], and Tony Tan[5]

[1] National Chengchi University, Taiwan
[2] RPTU Kaiserslautern-Landau, Germany
[3] Technical University of Munich, Germany
[4] Max Planck Institute for Software Systems, Germany
[5] University of Liverpool, United Kingdom

Abstract. Data sequences, such as time series, traces, and sensor streams, are words over large or infinite data domains. Their properties often depend on comparisons between values at different positions, which finite automata cannot express directly. We study synthesis and verification for Deterministic Register Automata (DRAs) over ordered data sequences. DRAs combine finite control with registers for remembered data values, providing compact symbolic descriptions of quantitative sequence languages. Our technical core is a local-search synthesis procedure and an exact robustness checker for DRAs. We fix the alphabet theory to ordered rationals $(\mathbb{Q}; <, =)$ with rational guard constants and use non-negative affine accumulator costs. For a fixed number of DRA registers, the checker runs in polynomial time; its dependence on that number is exponential. It reduces robustness to coverability and then to a shortest-path problem. Since the search ranges over $\mathbb{Q}^+$, it produces variable-length witnesses and, for RAA-definable alignment costs such as DTW, explores a counterexample space with no a priori length bound. As an application, we use learned DRAs as interpretable surrogates for queried neural sequence classifiers. A probably approximately correct (PAC) validation loop refines rejected candidates, returns genuine non-robustness witnesses for the classifier when found, and accepts a surrogate only with statistical guarantees for fidelity and local stability. We implement the framework in RAExc, enabling the synthesis of interpretable automata and the verification of their robustness. We evaluate the approach on benchmark languages and sequentialized MNIST, demonstrating high accuracy while verifying the robustness of the learned automata.

1 Introduction

Data sequences are words whose positions carry values from an infinite, or very large, data domain such as integers, rationals, or reals. They arise naturally in execution traces, sensor streams, event logs, financial indicators, and other time series data. Classical finite automata provide a canonical model of regular languages over finite alphabets, but they cannot directly express properties whose

truth depends on comparing data values across positions, such as whether successive peaks increase, whether future values exceed remembered thresholds, or whether a signal preserves a quantitative trend. This limitation motivates automata models over data words and data sequences, including data automata, nominal automata, and register automata [11,9,42,10,8]. Register automata extend finite control with finitely many registers that store selected input values and compare later inputs with those stored values or with constants. Over ordered domains such as $(\mathbb{Q}; <, =)$, they give a compact symbolic language for global quantitative patterns while retaining useful algorithmic properties, with applications in verification and synthesis [1,34,17].

This paper studies the synthesis and verification of deterministic register automata (DRAs) over numeric data sequences. On the synthesis side, we ask how to construct a small DRA from labelled sequence data or from a classifier that can be queried on examples. On the verification side, we ask whether the language recognised by a DRA is locally stable around a given data sequence under a quantitative perturbation cost. These two questions support one another: synthesis produces an interpretable symbolic model, and verification makes that model an object on which robustness questions can be answered exactly.

A timely source of such queried classifiers is modern sequence modelling with Recurrent Neural Networks (RNNs), LSTMs, and transformers [24,51,54]. These models are widely used on sequence data, but their internal decision procedures are difficult to inspect. A popular way to explain them is therefore to extract an automaton that approximates the trained network and then analyse that automaton instead of the original model. This idea has been studied for RNNs, weighted automata, and transformers [53,46,41,55,2]. Existing extraction methods, however, predominantly target automata over finite alphabets. This is a serious mismatch for numeric time series: discretising the values can obscure comparisons between positions, while a finite automaton cannot remember and compare data values in the way required by many natural quantitative rules. DRAs are a natural replacement because their registers provide precisely the finite memory needed for order and equality comparisons over data values.

For an automaton to be a useful explanation, agreement with the neural model is not enough. The explanation should also be stable under perturbations that are small according to the intended semantics of the data. If a surrogate changes its classification after an imperceptible or semantically insignificant edit, then it may describe a brittle boundary artefact rather than a robust pattern. We therefore seek *robust automata explanations*: compact DRAs that agree with the queried model on the inputs of interest and whose classifications persist through a neighbourhood of those inputs. The following example shows why numeric comparisons call for DRAs and why robustness matters.

Example 1. Consider daily closing prices for a market index, represented by a sequence $x_1, \ldots, x_n$. We use the DRA in Fig. 2 as the formal definition of the "higher highs and higher lows" indicator. It first identifies a trough and a following peak; thereafter, it requires the price to remain above the most recently confirmed trough until a new, higher trough is identified, and it then requires a

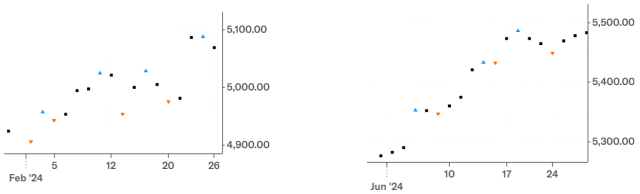


Fig. 1. Two S&P 500 price sequences. Red and blue triangles mark lows and highs, respectively. The left uptrend is robust to small last value decreases, while the right uptrend is fragile.

higher peak. Thus the DRA gives a streaming version of the usual higher-highs-higher-lows pattern, including a margin between confirmed extrema. Fig. 1 shows two historical S&P 500 price segments (Feb 1–26 and Jun 1–26, 2024). Both are classified as uptrends by this rule. However, under a small perturbation to the final price, the left sequence is *robust*: the signal flips only if the price drops below the previous low. The right sequence is *fragile*: even a slight drop can violate the DRA's running trough constraint and flip the signal. In practice, positive signals that are not robust warrant caution [44].

The robustness question raised by Example 1 exposes two limitations of existing robustness checkers for neural sequence models [36,16,31,43,56,39,49,38,27]. First, most RNN checkers propagate bounds or abstractions through specific recurrent cells, and transformer checkers reason about self-attention layers using convex relaxations, linear bounds, or path constraints. These methods can provide formal guarantees for the analysed neural architecture, but they require access to the model structure, parameters, and internal operations. Second, these checkers typically fix the shape of the input tensor before verification. For sequence models, this means that the input length is bounded in advance: the checker only perturbs the values of the original sequence.

This fixed-length view is appropriate for length-preserving L_p perturbations or for analyses with an explicit window bound, but it does not directly answer the language-theoretic question whether there exists *some* data sequence $w \in \mathbb{Q}^+$ in the neighbourhood of the original sequence that flips the label. The distinction matters for edit and alignment distances. Under edit costs, counterexamples may have lengths different from the original input. Under *dynamic time warping* (DTW) and related alignment costs, the length of a potential counterexample is not bounded by the original input length [45]. Thus, fixed-length neural checkers can miss the very counterexamples that sequence level robustness is intended to rule out.

Our approach is to verify robustness on the extracted DRA itself. Perturbation costs are specified by *Register Automata with Accumulator* (RAA), which scan a reference sequence and a candidate perturbed sequence while accumulating their distance. Given a DRA $\mathcal{A}$, a sequence v, and a cost function D represented by an RAA, local robustness asks whether $B_\delta(v) \cap L(\mathcal{A}^c) = \emptyset$, where $B_\delta(v) = \{w \in \mathbb{Q}^+ : D(v, w) < \delta\}$. The search ranges over all data sequences $w \in \mathbb{Q}^+$ rather than over a fixed-dimensional perturbation space. Consequently,

the checker can search for witnesses of variable length, and for costs definable by RAAs such as DTW and edit distance, it is not tied to any a priori bound on the counterexample length. For a fixed number of registers, we reduce this robustness problem to coverability for bounded projected RAAs and then to a single-source shortest path computation. When robustness fails, the same construction yields a concrete witness that flips the DRA label.

It remains to justify why the verified DRA explains the neural model. Our method uses the neural model only as a labelling oracle. It is therefore independent of the model architecture: the same extraction and validation loop can be applied to LSTMs, transformers, or any other classifier over numeric data sequences. The cost is that the guarantee transferred from the DRA to the neural model is statistical rather than absolute. This is the standard model-agnostic setting, where explanations are validated relative to a reference input distribution. Popular post-hoc explainers such as LIME, Anchors, and SHAP likewise characterise model behaviour through background samples or perturbation distributions [47,48,40]. Our loop learns a compact DRA from labelled samples, tests agreement with the neural model on fresh samples, and invokes the exact DRA robustness checker on samples where the two models agree. A failed robustness check either refines the learner or, if the neural model also flips on the returned witness, produces a genuine witness that the neural model is not robust. If the loop accepts, the DRA has PAC statistical guarantees for both fidelity and local stability under the reference distribution [33,5].

Contributions We make the following major contributions in this work.

- **Robustness verification for DRAs over data sequences.** We give a polynomial-time robustness checker for DRAs with a fixed number of registers over $(\mathbb{Q}; <, =)$, with perturbation costs specified by RAAs. Unlike fixed-length neural robustness checkers, the procedure searches over $w \in \mathbb{Q}^+$ and therefore supports variable-length counterexamples and, for metrics such as DTW, counterexample spaces of unbounded length.
- **Reduction to shortest paths.** We reduce DRA robustness to coverability for bounded projected RAAs and then to a single-source shortest- path problem. When robustness fails, the algorithm constructs a concrete label-flipping witness whose cost is within any prescribed $\epsilon > 0$ of the minimal cost (the minimum itself need not be attained under strict guards).
- **DRA synthesis from black-box sequence classifiers.** We develop an architecture-agnostic local search learner that synthesises DRAs from labelled samples obtained by querying a sequence classifier or generated from synthetic datasets.
- **PAC-validated robust explanations.** We combine learning and exact DRA robustness checking in a validation loop that refines rejected candidates, returns genuine non-robustness witnesses, or accepts a DRA with statistical guarantees for fidelity and local stability under a reference distribution.
- **Implementation and evaluation.** We implement the framework in *RAExc* and evaluate it on synthetic languages and sequentialised MNIST, demon-

strating that DRAs can approximate neural sequence classifiers and reveal non-robust behaviour in the learned surrogates.

Organisation. The remainder of this paper is organised as follows. Sec. 2 illustrates our automata learning and robustness verification procedures through a practical example. Sec. 3 introduces register automata, their variants, and the local robustness property. Sec. 4 presents an efficient robustness checking procedure for DRAs. Sec. 5 describes our learning algorithm, its integration with robustness checking, and the resulting statistical guarantees. Sec. 6 presents the experimental results. Finally, Sec. 7 discusses related work and concludes the paper.

2 Motivating Example

In this section, we elaborate on the higher-highs-higher-lows concept in Example 1 and offer an intuitive account of how our robustness verification and automata learning algorithms operate on it. A DRA embodying this concept is depicted in Fig. 2. To simplify our presentation, we focus on the case where the price decreases first (since the case of increasing first is symmetric), and normalise the price data by setting the initial price to zero.

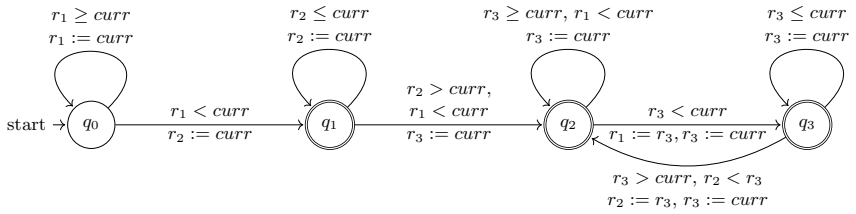


Fig. 2. A deterministic register automaton $\mathcal{A}$ that identifies the higher-highs-higher-lows price pattern.

The DRA $\mathcal{A}$ maintains three registers r_1, r_2, and r_3 (initialised to zero) representing the last trough, the last peak, and the previous value, respectively. It descends in q_0 until it reaches a trough, ascends in q_1 to a peak, and then alternates as follows: it remains in q_2 while the price stays above r_1; upon detecting a higher trough, it moves to q_3; and upon a higher peak, it returns to q_2. As an illustration, consider the price sequence $v := 0, -1, 5, 3, 7, 9, 6, 8$, which exhibits peaks at 5 and 9 and troughs at -1, 3, and 6. Since both the peaks and the troughs are strictly increasing, this sequence represents an uptrend and is therefore accepted by $\mathcal{A}$.

Robustness. Suppose we ask whether v remains an uptrend when its final value is perturbed by less than δ: does some $w \in \mathbb{Q}^+ \setminus L(\mathcal{A})$ satisfy $D(v, w) < \delta$, where $D((v_1, \ldots, v_n), (w_1, \ldots, w_n)) := |v_n - w_n|$ is the *last-letter* distance? We build an automaton $\mathcal{A}_{D,v}$ with an *accumulator* register: on reading the i-th letter w_i of $w = w_1, \ldots, w_n$, the transition taken adds $a_i w_i + b_i$ to the accumulator,

where the rationals a_i, b_i are fixed by that transition (and may depend on v), so that $\mathcal{A}_{D,v}(w) = \sum_{i=1}^{n}(a_i w_i + b_i)$. For the last-letter distance, $a_i = b_i = 0$ for $i < n$, while at $i = n$ the automaton takes $(a_n, b_n) = (-1, v_n)$ if $v_n > w_n$ and $(a_n, b_n) = (1, -v_n)$ otherwise, i.e. it adds $|v_n - w_n|$, giving $\mathcal{A}_{D,v}(w) = D(v, w)$.

Taking the product $\mathcal{A}'$ of $\mathcal{A}_{D,v}$ and $\mathcal{A}^c$ (the complement of $\mathcal{A}$) gives $\mathcal{A}'(w) = D(v, w)$ and $L(\mathcal{A}') = L(\mathcal{A}^c) \cap \mathbb{Q}^n$, reducing robustness to whether some $w \in L(\mathcal{A}')$ satisfies $\mathcal{A}'(w) < \delta$ — exactly what the procedure of Sec. 4 decides for general $\mathcal{A}$, v, and δ. Here, v admits an adversarial instance iff $\delta > 8 - 3 = 5$, since v stops being an uptrend once the last value 8 drops to 3 or below.

Learning. Suppose we have an accurate trading model $\mathcal{N}$ that classifies the two sequences in Fig. 1 as uptrends. Recall that the right-hand sequence is unstable under the last-letter distance: even an arbitrarily small perturbation ($\varepsilon > 0$) of the final price violates the relevant running constraint and flips the signal. For this sequence, our pipeline identifies a nearby sample as a witness of instability and refrains from certifying a robust DRA for $\mathcal{N}$. In contrast, the left-hand sequence is stable under the same metric—its signal remains unchanged unless the last price falls below the previous low. Consequently, we can learn a robust DRA for $\mathcal{N}$ over a neighbourhood of the left-hand sequence. By explicitly targeting robustness, the learned automaton captures the intended pattern with a margin: agreed sequences are locally stable, spurious counterexamples are filtered, and genuine non-robust witnesses are reported. This design turns the extracted DRA into an interpretable surrogate that provides both *agreement* and *stability* guarantees.

3 Preliminaries

We fix the ordered-rational alphabet theory $(\mathbb{Q}; <, =)$, with rational constants in guards. $\mathbb{Q}^+$ denotes the non-empty rational sequences; registers initially hold 0.

Register automata. A *k-register automaton* (k-RA) is $\mathcal{A} = (Q, \Delta, q_0, F)$, where Q is finite, $q_0 \in Q$, $F \subseteq Q$, and Δ is a finite set of transitions (p, φ, ψ, q). It has registers $\bar{r} = (r_1, \ldots, r_k)$. A guard $\varphi(\bar{r}, \mathit{curr})$ is a conjunction of atoms $x \sim y$, where $\sim \in \{<, \leq, =, \neq, \geq, >\}$ and $x, y \in \bar{r} \cup \{\mathit{curr}\} \cup \mathbb{Q}$. An assignment ψ is a parallel update in which every r_i receives a constant, a register value, or the current symbol.

A configuration is (p, ν) with $p \in Q$ and $\nu : \bar{r} \to \mathbb{Q}$. On reading a, the transition (p, φ, ψ, q) is enabled exactly when $\varphi(\nu, a)$ holds; it changes (p, ν) to $(q, \psi(\nu, a))$. A word is accepted if its run from $(q_0, \mathbf{0})$ ends in F; its language is $L(\mathcal{A})$. The RA is *deterministic* (a DRA) if at most one outgoing transition is enabled in every configuration. Missing transitions reject. We totalise a DRA, when needed, by adding a non-accepting sink, and then complement it by swapping final and non-final states. DRA languages are closed under Boolean operations [8]; write $\mathcal{A}^c$ for a DRA recognising $\mathbb{Q}^+ \setminus L(\mathcal{A})$.

Register automata with accumulators. A *k-register automaton with accumulator* (k-RAA) $\mathcal{A}_D = (Q, \Delta, q_0, F)$ reads a pair $(v, w) \in \mathbb{Q}^+ \times \mathbb{Q}^+$ with one head per sequence and computes a cost. Its transitions $(p, \varphi, \psi, \chi, q, \mathsf{mov})$ extend those of RAs: guards and assignments may refer to the letters $curr_1, curr_2$ under the two heads, $\mathsf{mov} \in \{1, 2, \mathsf{both}\}$ names the head(s) that advance, and $\chi : acc \mathrel{+}= a_1 \cdot curr_1 + a_2 \cdot curr_2 + b$ ($a_1, a_2, b \in \mathbb{Q}$) is an affine accumulator increment, required to be non-negative whenever φ holds (so that the shortest-path reduction of Sec. 4 has non-negative edges).

A configuration (p, i, j, ν) records the state, both head positions and the register valuation; the run starts in $(q_0, 1, 1, \mathbf{0})$. A transition is enabled if its guard holds and every head it reads or advances is not yet exhausted; a transition that ignores an exhausted head may still fire, which allows insertions and deletions at the end of a sequence. It updates the registers, advances the heads in mov (the others re-read their letter), and adds $\chi(v_i, w_j)$ to the cost. A run is accepting if it ends in a final state with both heads exhausted. The accumulator is write-only: guards never test it. RAAs are non-deterministic, which lets them choose, e.g., an alignment of v and w; $\mathcal{A}_D(v, w)$ is the minimum cost over all accepting runs (∞ if there is none), and $\mathcal{A}_D$ *computes* D if $\mathcal{A}_D(v, w) = D(v, w)$ for all (v, w). Registers store letters for later comparison, as in RAs; in Sec. 4, the registers of the DRA under analysis become registers of a product RAA. Define

$$wt(\mathcal{A}_D) = \inf_{v,w \in \mathbb{Q}^+} \mathcal{A}_D(v, w), \qquad wt(\mathcal{A}_D, v) = \inf_{w \in \mathbb{Q}^+} \mathcal{A}_D(v, w).$$

Robustness. For $\delta > 0$, let $B_\delta(v) = \{w \in \mathbb{Q}^+ : D(v, w) < \delta\}$. A DRA language $L(\mathcal{A})$ is δ-robust at $v \in L(\mathcal{A})$ iff $B_\delta(v) \cap L(\mathcal{A}^c) = \emptyset$. Hence robustness asks whether a word of cost below δ can reach the complement language. The RAAs computing Hamming, DTW, edit, Manhattan, and last-letter distance are given in App. A of [25].

4 Verifying Robustness of DRAs

In this section, we develop the decision procedure behind the following theorem.

Theorem 1. *For any fixed number of registers k and any fixed perturbation-cost RAA with non-negative updates, checking whether a k-DRA is robust at a given sequence is decidable in polynomial time. Moreover, whenever robustness fails, a concrete witness to non-robustness can also be constructed in polynomial time.*

The polynomial is in the automaton, the sequence, and the finite constant set induced by the instance. The graph constructed below has $O(|v||Q||C|^{k+1})$ vertices and $O(|v||\Delta||C|^{k+2})$ edges, so the running time of our algorithm is exponential in k. Some super-polynomial dependence on k is expected: robustness subsumes DRA emptiness, which is PSPACE-complete when k is part of the input [14], so no algorithm can be polynomial jointly in the automaton size and k unless P = PSPACE.

Our proof proceeds in two reductions: robustness of a k-DRA to coverability for a restricted class of RAA, *bounded projected RAA*, then coverability to single-source shortest paths. Both reductions run in polynomial time, and since all accumulator updates (hence all graph edges) are non-negative, Dijkstra's algorithm solves the shortest-path problem in polynomial time, giving Theorem 1.

For the first reduction, we use *coverability*, a thresholded *min-cost* question in the sense of cost-register automata over finite alphabets [3]; when accumulator updates are identically zero, it collapses to the usual reachability problem for register automata with linear arithmetic [12]. For an RAA $\mathcal{A}_D$, a sequence v, and a constant $\delta \in \mathbb{Q}$, we say that $\mathcal{A}_D$ is covered at v by δ if $wt(\mathcal{A}_D, v) < \delta$. The coverability problem asks, given $\mathcal{A}_D$, v, and δ, whether $\mathcal{A}_D$ is covered at v by δ. Robustness reduces to coverability: the product between $\mathcal{A}$ and $\mathcal{A}_D$ gives an RAA $\mathcal{A}'_D$ accepting (v, w) iff $w \notin L(\mathcal{A})$ and $(v, w) \in L(\mathcal{A}_D)$, so $L(\mathcal{A})$ is δ-robust at v iff $\mathcal{A}'_D$ is not covered at v by δ. The reduction can be strengthened so that $\mathcal{A}'_D$ is a *bounded projected RAA*: for some $\alpha > 0$, $\mathcal{A}'_D$ is *α-bounded* and *v-projected* if $\mathcal{A}'_D(u, w) \neq \infty$ implies $u = v$ and every letter d occurring in w satisfies $|d| \leq \alpha$. The formal reduction is the following lemma.

Lemma 1 (Reducing robustness to coverability). *Given a DRA $\mathcal{A}$, a sequence $v \in L(\mathcal{A})$, an RAA $\mathcal{A}_D$, and a rational number $\delta > 0$, we can construct in polynomial time a rational number $\alpha > 0$ and α-bounded v-projected RAA $\mathcal{A}'_D$ such that $L(\mathcal{A})$ is not δ-robust at v if and only if $\mathcal{A}'_D$ is covered at v by δ.*

Proof. Product. Restrict the first tape of $\mathcal{A}_D$ to v by conjoining $curr_1 = v_i$ to every transition that reads the i-th letter, tracking i in the control state; every increment then has the form $a \cdot curr_2 + b$. Totalise $\mathcal{A}$, let $\overline{\mathcal{A}}$ be its complement, and form the product in which $\overline{\mathcal{A}}$ moves exactly when the second head advances: a transition of $\mathcal{A}_D$ with $\mathsf{mov} = 1$ leaves the $\overline{\mathcal{A}}$-component unchanged, and one with $\mathsf{mov} \in \{2, \mathsf{both}\}$ is paired with a transition of $\overline{\mathcal{A}}$ whose guard and update are evaluated at $curr_2$. (A $\mathsf{mov} = 1$ transition rereads the current letter of w, which the DRA must read only once.) With final states $F_{\overline{\mathcal{A}}} \times F_{\mathcal{A}_D}$, the product $\mathcal{A}''_D$ accepts (v, w) with cost $\mathcal{A}_D(v, w)$ iff $\mathcal{A}_D$ accepts it and $w \notin L(\mathcal{A})$. It has polynomial size, non-negative increments, and the registers of both automata. Hence $L(\mathcal{A})$ is not δ-robust at v iff $\mathcal{A}''_D$ is covered at v by δ.
Bounding. Let C_0 contain 0, the letters of v, and the constants of $\mathcal{A}''_D$, and let

$$\alpha \coloneqq 1 + \max\big(\{|c| : c \in C_0\} \cup \{|b/a|, |(\delta - b)/a| : a\,curr_2 + b \text{ an increment, } a \neq 0\}\big).$$

On a run of cost below δ, each increment satisfies $0 \leq a\,curr_2 + b < \delta$, which confines every letter read with $a \neq 0$ to $[-\alpha + 1, \alpha - 1]$. After propagating register assignments, every other letter of w occurs only in atoms $w_i \sim w_j$ or $w_i \sim c$ with $c \in C_0$. Replacing these letters by rationals in $[-\alpha, \alpha]$ with the same order type relative to the fixed letters and C_0, which exist by density of $\mathbb{Q}$ (the margin $+1$ leaves room beyond all fixed values), preserves every guard and the cost. Hence conjoining $-\alpha \leq curr_2 \leq \alpha$ to every transition yields an α-bounded v-projected RAA $\mathcal{A}'_D$ that is covered at v by δ iff $\mathcal{A}''_D$ is.

For the second reduction, we show that minimal-cost witnesses can be found among sequences over a finite set of values. We need two auxiliary facts.

Lemma 2. *Let $\Pi \subseteq \mathbb{Q}^n$, let $\mathrm{cl}(\Pi)$ be its closure in $\mathbb{R}^n$, let $\delta > 0$, and let $f : \mathbb{R}^n \to \mathbb{R}$ be continuous. Then $f(x) < \delta$ for some $x \in \Pi$ iff $f(x) < \delta$ for some $x \in \mathrm{cl}(\Pi)$.*

Proof. The forward direction is immediate. Conversely, if $x \in \mathrm{cl}(\Pi)$ and $f(x) < \delta$, then by continuity $f < \delta$ on an open neighbourhood of x, which contains a point of Π.

Lemma 3. *Let $C \subseteq \mathbb{Q}$ contain 0, the constants of an RAA $\mathcal{A}_D$, and the letters of v. On every run of $\mathcal{A}_D$ on (v, w) with all letters of w in C, every register value lies in C.*

Proof. By induction along the run: registers start at $0 \in C$, and each assignment copies a constant, a register value, $curr_1$ (a letter of v), or $curr_2$ (a letter of w), all of which lie in C.

Lemma 4 (Reducing coverability to shortest path). *Let k be a fixed positive integer. Given α-bounded v-projected k-RAA $\mathcal{A}_D$ and a rational number $\delta > 0$, we can construct in polynomial time a weighted graph $G = (V, E)$, a source vertex $s \in V$ and a set $U \subseteq V$ of target vertices such that $\mathcal{A}_D$ is covered at v by δ iff there is a path in G with weight $< \delta$ from the source vertex s to some vertex in U.*[6]

Proof. Since $\mathcal{A}_D$ is v-projected, we write each increment as $a \times curr_2 + b$. Fix a transition sequence $\mathbf{t} = (t_1, \ldots, t_\ell)$ consuming n letters of w; as $\mathsf{mov} = 1$ transitions reread the current letter, t_i reads coordinate w_{n_i}, and several transitions may share one. Let $\Pi(\mathbf{t}) \subseteq \mathbb{Q}^n$ be the set of w on which $\mathbf{t}$ is an accepting run on (v, w). After substituting v and propagating register updates, every guard atom has the form $w_i \sim w_j$ or $w_i \sim c$, so $\Pi(\mathbf{t})$ is a (not necessarily closed) polytope, and the cost is the linear function $f_{\mathbf{t}}(w) = \sum_i (a_i w_{n_i} + b_i)$. Hence $\mathcal{A}_D$ is covered at v by δ iff $f_{\mathbf{t}}(w) < \delta$ for some $\mathbf{t}$ and $w \in \Pi(\mathbf{t})$, iff, by Lemma 2, the same holds for some $w \in \mathrm{cl}(\Pi(\mathbf{t}))$.

Let C consist of 0, the constants of $\mathcal{A}_D$, the letters of v, and $\pm\alpha$. As $\mathcal{A}_D$ is α-bounded, $\mathrm{cl}(\Pi(\mathbf{t}))$ is bounded, so $f_{\mathbf{t}}$ attains its minimum over it at a vertex w. Connect two coordinates of w whenever a constraint that is tight at w relates them. Every connected component contains a coordinate tied by a tight constraint to a constant in C: otherwise, shifting the whole component by $\pm\epsilon$ preserves all tight constraints and, for small ϵ, all others, so w would be the midpoint of two feasible points. Hence every coordinate of w lies in C, and by Lemma 3 so does every register value along the run.

G enumerates runs over values in C. A vertex (p, i, μ, z, τ) records the control state p, the first-head position i, the register valuation $\mu \in C^k$, the value $z \in$

[6] A weighted graph $G = (V, E)$ is a directed graph where every edge has a weight. The weight of a path is the sum of the weights of its constituent edges.

$C \cup \{\star\}$ given to the current letter of w ($\star$ if it has not been read yet, so that rereads by $\mathsf{mov} = 1$ transitions agree), and a tag τ that orders the infinitesimal offsets of values sharing a base value in C; the tag decides strict and disequality atoms at boundary values, and the number of tags depends only on k. Every transition of $\mathcal{A}_D$, together with a value in C for the current letter when $z = \star$, yields an edge if its guard holds under the tag, weighted by its non-negative increment. The source s is the initial configuration, and U contains the vertices with a final state, $i = |v| + 1$, and $z = \star$, i.e. both heads exhausted. The tag semantics is spelled out in App. B of [25].
Correctness and size. ($\Rightarrow$) Given an accepting run of cost below δ on $\mathbf{t}$, the minimising vertex w' of $\mathrm{cl}(\Pi(\mathbf{t}))$ has coordinates in C and $f_{\mathbf{t}}(w') < \delta$; the direction from w' into $\Pi(\mathbf{t})$ yields tags satisfying all strict atoms, so following $\mathbf{t}$ gives a path from s to U of weight $f_{\mathbf{t}}(w') < \delta$. ($\Leftarrow$) A path of weight below δ determines $\mathbf{t}$ and a point $w' \in \mathrm{cl}(\Pi(\mathbf{t}))$ with $f_{\mathbf{t}}(w') < \delta$; its tags give a rational perturbation into $\Pi(\mathbf{t})$, which by Lemma 2 keeps the cost below δ. Finally, G has $O(|v||Q||C|^{k+1})$ vertices and $O(|v||\Delta||C|^{k+2})$ edges, which is polynomial for fixed k.

Backtracking a shortest path yields, for every $\epsilon > 0$, a concrete witness whose cost is within ϵ of the minimal cost. The minimum itself need not be attained when strict guards are tight.

Proof (Proof of Theorem 1). By Lemma 1, non-robustness of $L(\mathcal{A})$ at v reduces in polynomial time to coverability of a bounded projected RAA with k plus the cost-RAA's registers, a fixed number. By Lemma 4, this reduces in polynomial time to a shortest-path query on a graph with non-negative edge weights, solved by Dijkstra's algorithm; backtracking the path yields the witness.

For cost functions such as Hamming distance, the witness w is forced to have the same length as v, so coverability can be solved by unrolling to that fixed length — an acyclic computation, polynomial in k as well. This fails once the constructed RAA permits witnesses of unbounded length (i.e. the unrolled computation may contain cycles), hence the graph construction above.

5 Synthesis of Robust DRAs

This section presents the full extraction pipeline: synthesising candidate DRAs and validating them as robust surrogates. Candidates are learned by a simulated-annealing heuristic, $\mathcal{L}_{\mathrm{LS}}$, accepting mutations with a temperature-dependent probability so score-decreasing moves can help the search escape local optima. At each round, a certification interface returns a genuine non-robustness witness, refines the labelled sample set, or accepts the candidate with statistical guarantees [33].

Synthesis Based on Simulated Annealing

For fixed numbers of states, registers, and constants, the candidate DRA space is finite. We search this space by simulated annealing: a candidate is scored by its classification accuracy on the labelled sample set $S = (S^+, S^-)$, optionally up-weighting a distinguished "golden" positive subset $G \subseteq S^+$,

$$\mathrm{score}(\mathcal{A}, S) = \frac{|S^+ \cap L(\mathcal{A})| + |S^- \setminus L(\mathcal{A})| + w\,|G \cap L(\mathcal{A})|}{|S^+| + |S^-| + w\,|G|}, \quad w = \begin{cases} \frac{|S^+|+|S^-|}{10\,|G|}, & G \neq \varnothing, \\ 0, & G = \varnothing. \end{cases}$$

The golden subset thus contributes a fixed tenth of the weight of S, irrespective of its size, and its elements are counted twice, once through S^+ and once through G; when $G = \varnothing$ the score reduces to plain classification accuracy.

The search space is finite because guards and assignments are drawn from finite menus. Fix a finite *constant pool* $K \subseteq \mathbb{Q}$. A *guard atom* is an interval constraint $\ell \bowtie_1 \mathit{curr} \bowtie_2 u$ with $\bowtie_1, \bowtie_2 \in \{<, \leq\}$ and bounds $\ell, u \in \bar{r} \cup K \cup \{\pm\infty\}$, and an assignment maps each register to an element of $\bar{r} \cup \{\mathit{curr}\} \cup K$; the *atom space* is the finite set of such guard–assignment pairs. At each step, the learner selects a weighted mutation: resampling a state's outgoing transitions from the atom space, flipping/optimising accepting states, or splitting a frontier transition to activate a fresh state; non-accepted moves are reverted in place, and a near-neutral split (activating a state while lowering the score by at most one sample's weight) is retained. Algorithm 1 gives the full procedure, keeping the best automaton $\mathcal{A}^*$ separate from the current $\mathcal{A}$. $\mathcal{L}_{\mathrm{LS}}$ is a heuristic: it returns the best-scoring DRA found within its budget, which need not be optimal and may be a local optimum. All guarantees we state for extracted DRAs come from the validation phase below, not from the search.

Algorithm 1 SimulatedAnnealing$(S, \mathcal{A}_0, T_0, \rho, T_{\min}, \mathit{max_iter}, \mathit{max_time})$

$\mathcal{A} \leftarrow \mathcal{A}_0$; $\mathcal{A}^* \leftarrow \mathcal{A}_0$; $T \leftarrow T_0$; $i \leftarrow 0$; $t_{\mathrm{start}} \leftarrow \mathrm{now}()$
while $i < \mathit{max_iter}$ **and** $\mathrm{elapsed}(t_{\mathrm{start}}) < \mathit{max_time}$ **do**
  $\mathcal{A}' \leftarrow$ Mutate$(\mathcal{A})$ ▷ weighted choice of the three operators above
  $\Delta \leftarrow \mathrm{score}(\mathcal{A}', S) - \mathrm{score}(\mathcal{A}, S)$
  if $\Delta \geq 0$ **or** $\mathrm{Bernoulli}(\exp(\Delta/T))$ succeeds **then**
    $\mathcal{A} \leftarrow \mathcal{A}'$ ▷ annealing acceptance
  else if $\mathcal{A}'$ activates a fresh state **and** $\Delta \geq -1/|S|$ **then**
    $\mathcal{A} \leftarrow \mathcal{A}'$ ▷ near-neutral split, retained
  end if
  if $\mathrm{score}(\mathcal{A}, S) > \mathrm{score}(\mathcal{A}^*, S)$ **then** $\mathcal{A}^* \leftarrow \mathcal{A}$
  $T \leftarrow \rho \cdot T$; **if** $T < T_{\min}$ **then** $T \leftarrow T_0$ ▷ reheat
  $i \leftarrow i + 1$
end while
return $\mathcal{A}^*$

PAC Validation and End-to-End Extraction

We couple simulated-annealing learning with black-box validation. The user fixes a target agreement threshold $p \in (0, 1)$, an accuracy tolerance $\gamma \in (0, 1)$ (the

slack allowed between p and the guaranteed agreement), and an error probability $\varepsilon \in (0,1)$. For each candidate DRA, draw $n = \lceil \ln(1/\varepsilon)/(2\gamma^2) \rceil$ fresh samples from the reference distribution $\mathcal{D}$ and permit at most $d_{\max} = \lfloor n(1-p) \rfloor$ disagreements with $\mathcal{N}$. On each agreeing sample, invoke the checker from Sec. 4. A label-flipping neighbour on which $\mathcal{N}$ also flips is returned as a genuine witness. The candidate is accepted iff it has at most $d_{\max}$ disagreements over the n samples and the checker finds no label-flipping neighbour whose label under $\mathcal{N}$ equals that of its centre; otherwise every such neighbour, together with the disagreement samples, refines the labelled set before the next synthesis iteration. An accepted candidate is a statistically faithful and locally stable surrogate in the sense of Theorem 2.

Equivalence Checking Exact equivalence against a black-box model with no equivalence oracle is generally unavailable, so our equivalence phase adopts a probably-approximately-correct (PAC) view [33]: replace the oracle with i.i.d. draws from the reference distribution $\mathcal{D}$, accept only if the empirical disagreement stays below a threshold, and lift this via a one-sided Hoeffding bound to a distribution-level fidelity guarantee. Accordingly, ACCEPT means PAC-validated fidelity under $\mathcal{D}$, not exact equivalence between $\mathcal{A}$ and $\mathcal{N}$.

Lemma 5 (PAC fidelity guarantee). *Let $p, \varepsilon, \gamma \in (0,1)$ and consider one validation phase for a candidate DRA $\mathcal{A}$, with $n = \lceil \ln(1/\varepsilon)/(2\gamma^2) \rceil$ and $d_{\max} = \lfloor n(1-p) \rfloor$. If the phase accepts $\mathcal{A}$, then, with probability at least $1-\varepsilon$ over its random sampling process, $\Pr_{w\sim\mathcal{D}}[\mathcal{N}(w) = \mathcal{A}(w)] \geq p - \gamma$.*

Proof. Condition on the candidate DRA selected before this validation phase. Let $X_i = \mathbf{1}[\mathcal{N}(w_i) \neq \mathcal{A}(w_i)]$ and $q = \Pr_{w\sim\mathcal{D}}[\mathcal{N}(w) \neq \mathcal{A}(w)]$. The X_i are i.i.d. Bernoulli variables with mean q. If the phase accepts, then $\frac{1}{n}\sum_{i=1}^{n} X_i \leq 1-p$. Hence, if $q > 1-p+\gamma$, acceptance implies $\frac{1}{n}\sum_{i=1}^{n} X_i - q < -\gamma$. Hoeffding's inequality therefore gives $\Pr[\text{the phase accepts}] \leq e^{-2n\gamma^2} \leq \varepsilon$. Thus the probability that the phase accepts while $q > 1-p+\gamma$ is at most ε. Equivalently, with probability at least $1-\varepsilon$, acceptance implies the stated agreement bound.

Robustness Checking The validation procedure checks local stability only on samples where $\mathcal{A}$ and $\mathcal{N}$ agree. For such a sample w, the checker from Sec. 4 searches $B_\delta(w)$ for a point on which $\mathcal{A}$ flips. If the network flips as well, the algorithm returns a genuine witness; otherwise the point is a refinement example for the learner. We write $\mathrm{Stab}_\delta(\mathcal{A}, w) \coloneqq \forall w' \in B_\delta(w) : \mathcal{A}(w') = \mathcal{A}(w)$.

Lemma 6 (PAC instability-mass bound on agreed inputs). *Assume the checker of Sec. 4 is complete for B_δ, that is, on every queried input w it reports some $w' \in B_\delta(w)$ with $\mathcal{A}(w') \neq \mathcal{A}(w)$ whenever such a w' exists. Let $\lambda \coloneqq \Pr_{w\sim\mathcal{D}}[\mathcal{N}(w) = \mathcal{A}(w) \wedge \neg\mathrm{Stab}_\delta(\mathcal{A}, w)]$. Suppose a validation phase checks $(\mathcal{N}, \mathcal{A})$ with sample size n. For any confidence parameter $\eta \in (0,1)$, if the phase accepts then, with confidence at least $1-\eta$, it holds that $\lambda \leq 1 - \eta^{1/n}$.*

Proof. Condition on the candidate DRA selected before this validation phase. For each sample, let Z_i indicate $\mathcal{A}(w_i) = \mathcal{N}(w_i) \wedge \neg\mathrm{Stab}_\delta(\mathcal{A}, w_i)$. These are i.i.d.

Bernoulli variables of mean λ. Checker completeness implies that acceptance entails $Z_i = 0$ for every i. If $\lambda > 1 - \eta^{1/n}$, the probability of the latter event is $(1-\lambda)^n < \eta$. Hence the probability that the phase accepts while $\lambda > 1 - \eta^{1/n}$ is below η, which proves the claim.

Combining Lemmata 5 and 6 with a union bound yields the following theorem.

Theorem 2 (End-to-end PAC guarantee for an accepted DRA). *Define* $\mathrm{Good}_\delta(\mathcal{N}, \mathcal{A}, w) \coloneqq (\mathcal{A}(w) = \mathcal{N}(w)) \wedge \mathrm{Stab}_\delta(\mathcal{A}, w)$. *Let* $n = \lceil \ln(1/\varepsilon)/(2\gamma^2) \rceil$ *be the sample size used by the validation procedure. For any* $\eta \in (0,1)$*, if it returns an accepted DRA* $\mathcal{A}$*, then with confidence at least* $1 - \varepsilon - \eta$*,*

$$\Pr_{w \sim \mathcal{D}}\big[\mathrm{Good}_\delta(\mathcal{N}, \mathcal{A}, w)\big] \geq p - \gamma - \big(1 - \eta^{1/n}\big).$$

Proof. By definition, with all probabilities taken over $w \sim \mathcal{D}$,

$$\Pr[\mathrm{Good}_\delta(\mathcal{N}, \mathcal{A}, w)] = \Pr[\mathcal{A}(w) = \mathcal{N}(w)] - \Pr[\mathcal{A}(w) = \mathcal{N}(w) \wedge \neg \mathrm{Stab}_\delta(\mathcal{A}, w)].$$

With confidence at least $1-\varepsilon$, Lemma 5 lower-bounds the first term by $p-\gamma$; with confidence at least $1 - \eta$, Lemma 6 bounds the second term above by $1 - \eta^{1/n}$. A union bound yields the stated result.

Remark 1. The theorem is a guarantee about $\mathcal{A}$, not $\mathcal{N}$: $\mathrm{Stab}_\delta(\mathcal{A}, w)$ holds for the *surrogate*, but $\mathcal{N}$ may still flip somewhere in $B_\delta(w)$ where $\mathcal{A}$ does not. It also concerns one validation phase, whereas extraction performs up to B of them and returns the first acceptance; since each phase draws its n samples afresh, independent of the candidate, a union bound over the B phases gives Theorem 2 at confidence $1 - B(\varepsilon + \eta)$, restored to the stated confidence by running each phase at $\varepsilon/B, \eta/B$, costing only $\ln B/(2\gamma^2)$ extra samples per phase.

6 Experimental Evaluation

We evaluate *RAExc* [26], which extracts deterministic register automata (DRAs) as interpretable surrogates for neural sequence classifiers. It combines local-search DRA synthesis with fixed-register robustness checking under last-letter, edit, Hamming, Manhattan, and DTW distance through the RAA construction. We focus on the following research questions:

- **(RQ1)** Do the benchmark languages admit compact DRAs, and are the networks accurate on them?
- **(RQ2)** How effective is the PAC validation loop at accepting surrogates or finding robustness violations?
- **(RQ3)** How do learning and checking scale with the numbers of states and registers and with the input length?

Benchmarks. Table 1 lists the benchmarks. L_1–L_7 are the Tomita-style languages [53] with a and b encoded as distinct data values, including counting modulo a constant (L_5, L_6). S_1–S_{11} are idealised trend patterns from technical analysis [44], whose membership depends on comparing values at different positions; the learner is given one (S_1–S_8) or two (S_9–S_{11}) registers. MNIST reads each image as the sequence of its row-mean intensities and has no ground-truth DRA.

Table 1. Benchmarks and RQ1 results. L_i: Tomita languages; S_i: trend patterns over rational sequences, incl. the higher-highs/higher-lows family of Example 1 (S_9–S_{11}). *Acc*: balanced test accuracy (%); *Time*: synthesis time (s); *S/R/T*: states/registers/-transitions.

		$\mathcal{N}_{\text{LSTM}}$	$\mathcal{N}_{\text{T}}$	$\mathcal{L}_{\text{LS}}$				
Language	**Description**	**Acc**	**Acc**	**Acc**	**Time**	**S**	**R**	**T**
L_1	a^*	100.00	100.00	100.00	1.00	2	1	2
L_2	$(ab)^*$	100.00	100.00	99.95	20.08	4	1	8
L_3	$a^n b^m$ with odd n and even m	100.00	100.00	100.00	0.53	4	1	4
L_4	No symbol repeats three times consecutively	97.90	99.30	99.45	27.90	4	2	8
L_5	$a(a\|b)^*$ with even $\text{count}_a, \text{count}_b$	100.00	100.00	79.00	753.26	4	1	7
L_6	$a(a\|b)^*$ with $\text{count}_a \equiv_3 \text{count}_b$	100.00	100.00	100.00	11.72	3	2	6
L_7	$a^+ b^* a^* b^*$	100.00	98.20	100.00	335.81	5	1	10
S_1	Strictly increasing	99.80	99.65	100.00	2.06	2	1	3
S_2	Strictly decreasing	99.80	99.30	100.00	39.85	2	1	2
S_3	Non-strictly increasing	100.00	100.00	100.00	0.30	2	1	3
S_4	Non-strictly decreasing	99.90	100.00	100.00	2.66	2	1	4
S_5	Single peak	100.00	99.35	100.00	353.90	4	1	8
S_6	Single valley	100.00	98.30	100.00	310.51	4	1	7
S_7	Two peaks	99.95	97.75	99.80	262.98	5	1	10
S_8	Three peaks	100.00	99.65	100.00	83.02	6	1	12
S_9	Higher highs/lows	99.65	99.80	99.90	543.55	5	2	10
S_{10}	Higher highs, lower lows	99.55	98.75	99.60	504.28	5	2	10
S_{11}	Lower highs/lows	100.00	100.00	99.40	387.35	5	2	10
MNIST	Row-mean pixel intensity, digit 1 vs. rest	98.63	98.50	95.37	832.09	4	1	7

Experimental Setup. We use $p = 0.95$ and $\varepsilon = \gamma = 0.05$ ($n = 600$, $d_{\max} = 30$), strict neighbourhoods with $\delta = 1$ ($\delta = 1.5$ for Hamming and edit distance), and per-configuration budgets of 30 minutes for learning and 60 minutes for checking. Further details, including neural network training and the local-search hyperparameters, are given in App. C.1 of [25].

RQ1: Language identifiability. Table 1 shows that $\mathcal{L}_{\text{LS}}$ finds DRAs with 2–6 states and 1–2 registers reaching at least 99.4% on all synthetic benchmarks but L_5 (79.0%) and 95.37% on MNIST. We compare $\mathcal{L}_{\text{LS}}$ with two alternatives in App. C.4 of [25]. An SMT-based learner, given the same samples and budget, reaches 91.05–100% on L_1–L_7 and S_1–S_8, as it stops at 90–95% training accuracy, is slower on most of them (median 82 s vs. 28 s), and times out on the two-register patterns S_9–S_{11}. The active learner RALib [15] instead queries the language: it learns L_1–L_7 and S_1–S_8 exactly within 30 s, but times out or fails on S_9–S_{11} and cannot run on MNIST, which has no membership oracle. Thus

only $\mathcal{L}_{\mathrm{LS}}$ handles all benchmarks. Both networks reach at least 97.75% on every synthetic benchmark and 98.5% on MNIST, so RQ2 is not confounded by poorly trained black boxes.

RQ2: End-to-end robustness checking. For each of the 190 combinations of benchmark, network and metric, we run the full validate–check–refine loop of Sec. 5, not a single checker call. A run ends *robust* when a surrogate is PAC-accepted, *non-robust* when a witness also flips the network, and *inconclusive* after 10 rounds otherwise. 180 runs concluded (38 robust, 142 non-robust, no timeouts) with a median of 186 s and at most 1659 s; learning took 92.0% of the runtime and checking 2.8%; cf. App. C.5 of [25].

Table 2. Validation loop per metric (38 configurations each). *Multi*/*Max*: concluded runs with several rounds / most rounds; *Spur.*: spurious witnesses in concluded runs; *Cost*: confirmed witness costs; $|w| \neq |v|$: witnesses changing the length.

Metric	$+/-/?$	Multi	Max	Spur.	Cost	$\lvert w\rvert \neq \lvert v\rvert$
Last letter	21/14/3	8	5	358	0.0625	0
Edit	0/36/2	6	6	362	1	5
Hamming	2/35/1	5	5	50	1	0
Manhattan	15/19/4	11	9	835	0.011–0.106	0
DTW	0/38/0	6	5	232	0–0.083	36

Table 2 details the loop. 36 concluded runs needed refinement, which discarded 1,837 spurious witnesses that flip the DRA but not the network. Accepted surrogates agree with the network on at least 95.2% (median 99.6%) of the validation samples. Of the 10 inconclusive runs, seven still exceeded $d_{\max}$ after 10 rounds and three kept producing spurious witnesses.

Robustness holds mainly under in-place perturbations: last-letter and Manhattan distance account for 36 of the 38 accepted surrogates, and their witnesses cost at most 0.11. One substitution or edit flips the network in 35 and 36 of 38 configurations, respectively. DTW exposes non-robustness in all 38: 26 witnesses cost 0, so realignment alone flips the label, and 36 change the input length, which fixed-length checkers cannot detect.

RQ3: Scalability. We use three families: strictly increasing sequences of length n (one register, $n+1$ states, $n \in \{20, 40, \ldots, 100\}$); length-$n$ sequences of pairwise distinct values (n registers, $n+2$ states, $n \in \{2, 4, 6\}$); and S_1 with maximum training length 20–100. For each point, $\mathcal{L}_{\mathrm{LS}}$ learns a DRA of the stated size, which we test on 1,000 and check on 100 fresh sequences under last-letter distance ($\delta = 1$).

As shown in Fig. 3, learning is the bottleneck. For increasing sequences, it exhausts the budget from 20 states on, yet reaches 94.7–98.4% test accuracy; for distinct values, its time grows from 6 s to 1029 s between 2 and 6 registers; input length hardly matters (every S_1 point within 9 s, at least 99.5% accuracy). Checking stays cheap, at most 11 s per sequence: its time grows roughly linearly with the number of states (0.27–3.37 s), mildly with input length (0.08–0.43 s), and not with registers (0.01 s).

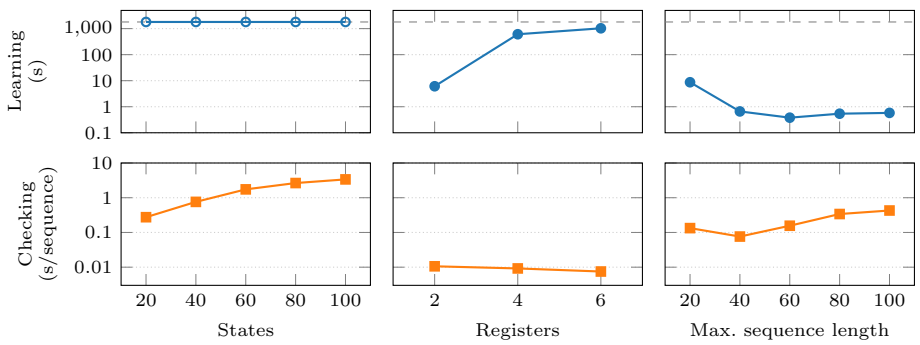


Fig. 3. Learning time and checking time per sequence under last-letter distance.

7 Related Work and Conclusion

Automata learning underlies grammatical inference, from graph-colouring and SAT-based DFA identification [23,22] to EDSM-style state merging [21,37]. Related register-automata learners [15,42,6] do not learn constants in transition guards, which our benchmarks require. Formal explainability studies complexity and constraint-based explanations [7,50], while extraction methods derive DFAs, weighted automata, or Moore machines from RNNs and transformers [19,53,4,46,52,2]. These use finite alphabets; we learn DRAs over data domains. For robustness, solver and abstraction methods [32,18,36,16,31,43,56,39,49,38,27,28] are mostly white-box, whereas probabilistic approaches [35] match our black-box validation setting. Our counterexamples are minimum-change contrastive explanations, in contrast to subset-minimal abductive or contrastive explanations [29,30,7].

Beyond DRAs, symbolic register automata [13], data automata [9,17], and variable automata [17,20] suggest richer future surrogates, for instance for LRA guards in financial indicators [44]. In summary, we give a polynomial-time robustness checker for DRAs with a fixed number of registers and combine it with local-search learning and PAC validation into a black-box pipeline that extracts compact surrogates and non-robustness witnesses for sequence classifiers.

Several limitations remain. The polynomial bound assumes a fixed number of registers; scaling to many registers needs better symbolic reductions. The validation guarantee is distributional: an accepted DRA is faithful on the reference distribution, not on all sequences. The learner is heuristic and may fail when the network has no compact DRA approximation. Natural future work includes probabilistic and nondeterministic surrogates, richer LRA-based guards, and properties such as fairness, monotonicity, and temporal specifications.

Acknowledgments. Jiang, Lin, Markgraf, and Parsert are supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant number 101089343. Chih-Duo Hong is supported by the National Science and Technology Council, Taiwan, under grant numbers 112-2222-E004-001-MY3 and 114-2634-F-004-002-MBK. We thank Micha Schrader for the discussion, anonymous SETTA 2026 reviewers for helpful comments and suggestions.

References

1. Abdulla, P.A., Atig, M.F., Kara, A., Rezine, O.: Verification of dynamic register automata. In: FSTTCS. pp. 653–665 (2014)
2. Adriaensen, R., Maene, J.: Extracting moore machines from transformers using queries and counterexamples. In: IDA. pp. 419–431 (2025)
3. Alur, R., D'Antoni, L., Deshmukh, J., Raghothaman, M., Yuan, Y.: Regular functions and cost register automata. In: LICS. pp. 13–22 (2013)
4. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
5. Angluin, D.: Queries and concept learning. Mach. Learn. **2**, 319–342 (1988)
6. Balachander, M., Filiot, E., Gentilini, R.: Passive learning of regular data languages in polynomial time and data. In: CONCUR. pp. 10:1–10:21 (2024)
7. Barceló, P., Monet, M., Pérez, J., Subercaseaux, B.: Model interpretability through the lens of computational complexity. NeurIPS **33**, 15487–15498 (2020)
8. Bojanczyk, M.: Slightly infinite sets. `www.mimuw.edu.pl/~bojan/upload/main-10.pdf` (2019)
9. Bojanczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data words. ACM Trans. Comput. Log. **12**(4), 27:1–27:26 (2011)
10. Bojanczyk, M., Klin, B., Lasota, S.: Automata theory in nominal sets. Log. Methods Comput. Sci. **10**(3) (2014)
11. Bojanczyk, M., Muscholl, A., Schwentick, T., Segoufin, L., David, C.: Two-variable logic on words with data. In: LICS. pp. 7–16 (2006)
12. Chen, Y.F., Lengál, O., Tan, T., Wu, Z.: Register automata with linear arithmetic. In: LICS. pp. 1–12 (2017)
13. D'Antoni, L., Ferreira, T., Sammartino, M., Silva, A.: Symbolic register automata. In: CAV. pp. 3–21 (2019)
14. Demri, S., Lazic, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. **10**(3), 16:1–16:30 (2009)
15. Dierl, S., Fiterau-Brostean, P., Howar, F., Jonsson, B., Sagonas, K., Tåquist, F.: Scalable tree-based register automata learning. In: TACAS. pp. 87–108 (2024)
16. Du, T., Ji, S., Shen, L., Zhang, Y., Li, J., Shi, J., Fang, C., Yin, J., Beyah, R., Wang, T.: Cert-RNN: Towards certifying the robustness of recurrent neural networks. In: CCS. pp. 516–534 (2021). `https://doi.org/10.1145/3460120.3484538`
17. Figueira, D., Lin, A.W.: Reasoning on data words over numeric domains. In: LICS. pp. 37:1–37:13 (2022)
18. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.T.: AI2: safety and robustness certification of neural networks with abstract interpretation. In: IEEE S&P. pp. 3–18 (2018)
19. Giles, C.L., Miller, C.B., Chen, D., Chen, H.H., Sun, G.Z., Lee, Y.C.: Learning and extracting finite state automata with second-order recurrent neural networks. Neural Comput. **4**(3), 393–405 (1992)
20. Grumberg, O., Kupferman, O., Sheinvald, S.: Variable automata over infinite alphabets. In: LATA. pp. 561–572 (2010)
21. Heule, M., Verwer, S.: Software model synthesis using satisfiability solvers. Empir. Softw. Eng. **18**(4), 825–856 (2013)
22. Heule, M.J., Verwer, S.: Exact dfa identification using sat solvers. In: ICGI. pp. 66–79 (2010)
23. de la Higuera, C.: Grammatical Inference: Learning Automata and Grammars. Cambridge University Press (2010)

24. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735
25. Hong, C.D., Jiang, H., Lin, A.W., Markgraf, O., Parsert, J., Tan, T.: Synthesis and verification of register automata over data sequences (technical report) (2026), https://arxiv.org/abs/2511.19100
26. Hong, C.D., Jiang, H., Lin, A.W., Markgraf, O., Parsert, J., Tan, T.: Synthesis and verification of transformer programs (Oct 2026). https://doi.org/10.5281/zenodo.23451085, https://doi.org/10.5281/zenodo.23451085
27. Hong, C.D., Yang, C.C., Wang, Y., Yu, F.: Influence-guided concolic testing of transformer robustness. arXiv:2509.23806 (2025). https://doi.org/10.48550/arXiv.2509.23806
28. Huang, M.I., Hong, C.D., Yu, F.: Concolic testing on individual fairness of neural network models. arXiv:2509.06864 (2026)
29. Ignatiev, A., Narodytska, N., Marques-Silva, J.: Abduction-based explanations for machine learning models. In: AAAI. pp. 1511–1519 (2019)
30. Ignatiev, A., Narodytska, N., Marques-Silva, J.: From contrastive to abductive explanations and back again. In: AIxIA. pp. 335–355 (2021)
31. Jacoby, Y., Barrett, C., Katz, G.: Verifying recurrent neural networks using invariant inference. In: ATVA (2020)
32. Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient SMT solver for verifying deep neural networks. arXiv:1702.01135 (2017)
33. Kearns, M.J., Vazirani, U.: An introduction to computational learning theory. MIT press (1994)
34. Khalimov, A., Kupferman, O.: Register-bounded synthesis. In: CONCUR. pp. 25:1–25:16 (2019)
35. Khmelnitsky, I., Neider, D., Roy, R., Xie, X., Barbot, B., Bollig, B., Finkel, A., Haddad, S., Leucker, M., Ye, L.: Analysis of recurrent neural networks via property-directed verification of surrogate models. STTT **25**(3), 341–354 (2023)
36. Ko, C., Lyu, Z., Weng, T., Daniel, L., Wong, N., Lin, D.: POPQORN: Quantifying robustness of recurrent neural networks. In: ICML. pp. 3468–3477 (2019)
37. Lang, K.J., Pearlmutter, B.A., Price, R.A.: Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. In: Grammatical Inference. pp. 1–12 (1998)
38. Liao, B.H., Cheng, C., Esen, H., Knoll, A.: Are transformers more robust? towards exact robustness verification for transformers. In: SafeComp (2023). https://doi.org/10.1007/978-3-031-40923-3_8
39. Lin, L.J., Hong, C.D.: Robustness verification of recurrent neural networks with abstraction refinement. arXiv:2606.12490 (2026)
40. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: NeurIPS. pp. 4765–4774 (2017)
41. Merrill, W., Tsilivis, N.: Extracting finite automata from rnns using state merging. arXiv:2201.12451 (2022)
42. Moerman, J., Sammartino, M., Silva, A., Klin, B., Szynwelski, M.: Learning nominal automata. In: POPL. pp. 613–625 (2017)
43. Mohammadinejad, S., Paulsen, B., Wang, C., Deshmukh, J.V.: DiffRNN: Differential verification of recurrent neural networks. arXiv:2007.10135 (2020)
44. Murphy, J.J.: Technical analysis of the financial markets. New York Institute of Finance (1999)
45. Nielsen, A.: Practical Time Series Analysis: Prediction with Statistics and Machine Learning. O'Reilly (2019)

46. Okudono, T., Waga, M., Sekiyama, T., Hasuo, I.: Weighted automata extraction from recurrent neural networks via regression on state spaces. In: AAAI. pp. 5306–5314 (2020)
47. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. In: KDD. pp. 1135–1144 (2016)
48. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: AAAI. pp. 1527–1535 (2018)
49. Shi, Z., Zhang, H., Chang, K., Huang, M., Hsieh, C.: Robustness verification for transformers. In: ICLR (2020). https://doi.org/10.48550/arXiv.2002.06622
50. Shrotri, A.A., Narodytska, N., Ignatiev, A., Meel, K.S., Marques-Silva, J., Vardi, M.Y.: Constraint-driven explanations for black-box ML models. In: AAAI. pp. 8304–8314 (2022)
51. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)
52. Wei, Z., Zhang, X., Sun, M.: Extracting weighted finite automata from recurrent neural networks for natural languages. arXiv:2206.14621 (2022)
53. Weiss, G., Goldberg, Y., Yahav, E.: Extracting automata from recurrent neural networks using queries and counterexamples. In: ICML. pp. 5244–5253 (2018)
54. Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., Sun, L.: Transformers in time series: A survey. In: IJCAI. pp. 6778–6786 (2023)
55. Zhang, X., Du, X., Xie, X., Ma, L., Liu, Y., Sun, M.: Decision-guided weighted automata extraction from recurrent neural networks. In: AAAI. pp. 11699–11707 (2021)
56. Zhang, Y., Du, T., Ji, S., Tang, P., Guo, S.: RNN-Guard: Certified robustness against multi-frame attacks for recurrent neural networks. arXiv:2304.07980 (2023)