Benchmarking State-Merging Algorithms for Learning Regular Languages

Adil Soubki ASOUBKI@CS.STONYBROOK.EDU
Department of Computer Science
Institute for Advanced Computational Science
Stony Brook University

Jeffrey Heinz JEFFREY.HEINZ@STONYBROOK.EDU
Department of Linguistics
Institute for Advanced Computational Science
Stony Brook University

Editors: François Coste, Faissal Ouardi and Guillaume Rabusseau

Abstract

The state-merging algorithms RPNI, EDSM, and ALERGIA are tested on MLRegTest, a benchmark for the learning of regular languages (van der Poel et al., 2023). MLRegTest contains training, development, and test data for 1,800 regular languages, which themselves are from several well-studied subregular classes. The results show that there is large variation in the performance of these algorithms on the benchmark with EDSM performing the best overall. Furthermore, the mean accuracies on the test data for all three state-merging algorithms are less than the mean accuracies obtained by the neural networks van der Poel et al. (2023) studied. A further experiment augments the training data in MLRegtest with shorter strings and shows they dramatically improve the performance of the state-merging algorithms.

1. Introduction

This paper presents the results of three well-known state-merging algorithms on MLRegTest, a benchmark for the machine learning of sequential classifiers (van der Poel et al., 2023) MLRegTest contains training, development, and test sets from 1,800 regular languages organized into classes spanning several subregular regions. The three algorithms tested are RPNI (Oncina and García, 1992), EDSM (Lang et al., 1998), and ALERGIA (Carrasco and Oncina, 1994, 1999). de la Higuera (2010) provides a uniform presentation of these algorithms.[1] The results of this investigation reveal that while there is large variation in the performance of these algorithms on the benchmark, EDSM overall performs the best. However, the mean accuracies on the test data for all three state-merging algorithms are less than the mean accuracies obtained by the neural networks van der Poel et al. (2023) studied.

To some extent this is not too surprising given that the parameters of the neural networks numbered in the hundreds of thousands, and the state merging algorithms are much simpler.

1. RPNI is an acronynm for "Regular Positive and Negative Inference." EDSM is an acronym for "Evidence Driven State Merging." Neither Carrasco and Oncina (1994) (who named the algorithm) nor de la Higuera (2010) explain whether ALERGIA is an acronym even though it is common in grammatical inference for 'IA' to abbreviate "Inference Algorithm."

On the other hand, there are theoretical results for RPNI and ALERGIA (see §2) which would lead one to expect better performance.

We also observe that the shortest training strings in MLRegTest are of length 20. It may be that shorter strings are necessary for the state-merging algorithms to do well on the languages in MLRegTest. In the parlance of grammatical inference, characteristic samples for these languages may necessarily contain strings shorter than length 20. We therefore conducted additional experiments where the training sets in MLRegTest were augmented with fewer than 1,000 shorter strings and ran the state-merging algorithms again. The results are encouraging for EDSM and RPNI, which see a nearly 15 point improvement in accuracy. Obviously, it is important to also examine the impact of shorter strings on the performance of neural networks, a task left for future research.

All of the code used, data sets developed, and observations collected in this course of this research are freely available on GitHub.[2].

2. Learning Regular Languages with State Merging

This section provides an overview of RPNI, EDSM, and ALERGIA. Readers are referred to de la Higuera (2010) for details and Heinz et al. (2015) for additional discussion. RPNI, EDSM, and ALERGIA are state-merging algorithms. These algorithms and many other variants are implemented in the FlexFringe software package, which provides a customisable toolkit for state-merging (Verwer and Hammerschmidt, 2022, 2017).

RPNI and EDSM take as input a set of strings, each of which is labeled as to whether or not they belong to the target language, and outputs a finite state acceptor (FA) which classifies strings accordingly. ALERGIA takes as input a similar set of labeled strings and outputs a probabilistic finite state acceptor (PFA) which assigns probabilities to strings. To use ALERGIA for sequential classification when the training data contains both positive and negative examples, as we do here, the algorithm is applied to the positive strings to output a PFA for the positive strings P, and it is applied separately to the negative strings to output a PFA for the negative strings N. These two machines together provide a classifier: given some string x, $P(x)$ is compared to $N(x)$ and x is classified according to which is more likely.

A prefix tree of these input strings is constructed and the states are either left unmarked, or labeled as accepting or rejecting. Subsequently, a search is conducted for pairs of states to merge. RPNI, EDSM, and ALERGIA differ in both how this search is conducted and the decision procedure used to determine if a given pair of states should be merged. We first illustrate the prefix tree, then explain state-merging, and finally discuss particulars of the search.

Figure 1a presents an example prefix tree for a data sample where the positive strings are $\{aa, b\}$ and the negative strings are $\{ba\}$. In the figure, state 1 is the starting state and states are labeled A for accepting and R for rejecting. To illustrate state-merging, consider Figure 1b where states 2 and 3 in the prefix tree above have been merged. When two states are merged, the transitions must be preserved. The transition from state 2 to 3 in Figure 1a is now a self-loop in Figure 1b. This is how generalization can occur. Whereas the prefix tree recognizes only finitely many strings, the acceptor in Figure 1b recognizes infinitely

2. https://github.com/adil-soubki/mlrt

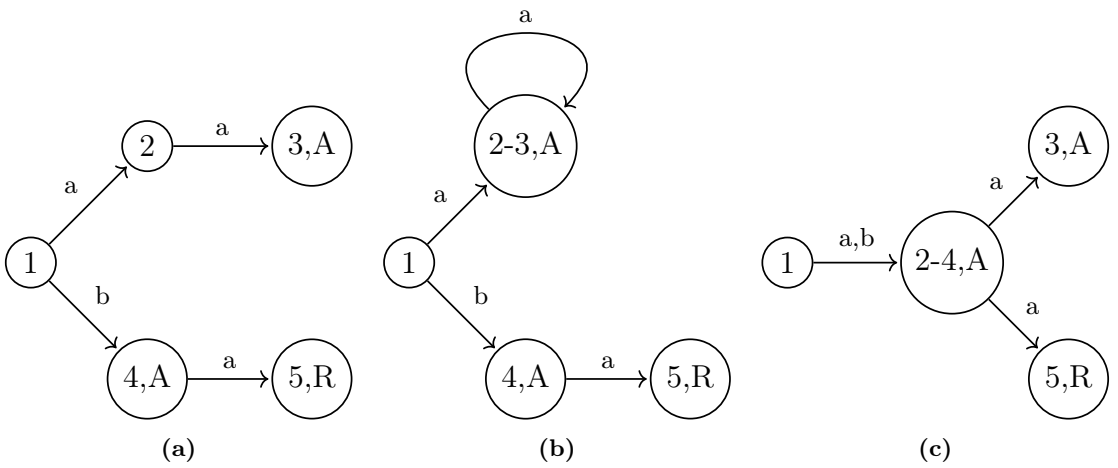

Figure 1: **(a)** The prefix tree built from positive strings are $\{aa, b\}$ and the negative strings are $\{ba\}$. **(b)** The acceptor obtained by merging states 2 and 3 in the prefix tree. **(c)** The acceptor obtained by merging states 2 and 4 in the prefix tree.

many. Furthermore, the merged state is accepting because it inherits that property from state 3. Similarly, if state 2 was merged with state 5 the merged state would be rejecting because that property is inherited as well. However, it is not possible to merge an accepting state with a rejecting state because those two properties are incompatible: a state cannot be both accepting and rejecting.

When two states are merged, non-determinism can occur. Figure 1c shows the machine obtained by merging states 2 and 4 in the prefix tree, which introduces non-determinism. RPNI, EDSM, and ALERGIA resolve non-determinism recursively by proceeding to merge states to eliminate any introduced non-determinism. In this example, these algorithms would next attempt to merge states 3 and 5 to eliminate the non-determinism. However, this attempt would fail because 3 is accepting and 5 is rejecting. Consequently, the merging of states 2 and 4 would be rejected as well.

It is well known that for any regular language L, there exists a finite positive sample S such that there is a way to merge states in the prefix tree T of S to yield an acceptor A which recognizes L exactly (Heinz et al., 2015, §3.4.4). The problem is knowing which states to merge. So how do RPNI, EDSM, and ALERGIA merge states?

RPNI, EDSM, and ALERGIA have procedures which tell them which states to try to merge. If they can merge them they do, but if they cannot (because they lead to the merging of incompatible states) then they conclude that those two states must be distinct. These procedures can be understood in terms of state coloring, a convention used to keep track of the state of search (de la Higuera, 2010). A state is colored red if it will be included in the final automaton. Initially, only the initial state of the prefix tree is colored red. A state is colored blue if it is a candidate state to be merged with some red state. Other states are colored white. White states will eventually be colored blue, and then red.

The algorithms will attempt to merge each blue state q with each red state. If q can merge with some red state, it is merged and that state stays red. If q cannot merge with

any red state, then it becomes colored red and its successor states which are not red are colored blue. State merging stops when all states are red.

RPNI conducts a breadth-first search of the prefix tree when trying to merge states. So given two potential merges – blue state b with red state r or blue state b' with red state r' – it will always choose the pair with blue state closest to the initial state. If b and b' are equally close, it will choose the pair whose red state is closest to the initial state. EDSM instead computes a score for all red and blue state pairs and first tries to merge the pair with the highest score. The score of a merge is determined by the number of strings in the sample that would end in the merged state. ALERGIA uses the Hoeffding statistical test to decide whether to merge states. The states are merged if the Hoeffding test determines that the frequency distribution over the symbols on the outgoing transitions of the two states are not statistically significantly different.

To conclude this section, we summarize the theoretical and empirical results of these algorithms. RPNI provably identifies in the limit the class of regular languages from positive and negative data and runs in polynomial time and data (Oncina and García, 1992). Theoretical results are not known for EDSM but it outperformed other state merging algorithms in the Abbadingo One competition (Lang et al., 1998). ALERGIA provably identifies any deterministic PFA in the limit with probability one and runs in time polynomial in the size of the sample (Carrasco and Oncina, 1999; de la Higuera and Thollard, 2000).

3. Experimental Setup

This section provides a discussion of notable implementation details for the algorithms (§3.1), as well as an overview of the data set chosen for this study, MLRegTest (§3.2), and concludes with a description of our experimental design and evaluation process (§3.3).

3.1. Implementation Details

The implementations of RPNI and ALERGIA provided by FlexFringe take minor departures from those described in the original papers. For RPNI, the original algorithm considers merging blue states in length-lexicographic order while FlexFringe instead uses a score when blue/red state pairs are equally distant from the initial state. Similarly, note that the version of ALERGIA we utilize is very similar to the original described by Carrasco and Oncina (1994), but with a few small changes as discussed by Verwer and Hammerschmidt (2022, §4.1).

3.2. MLRegTest

We train and test our models using MLRegTest (van der Poel et al., 2023), a benchmark data set for the learning of regular languages. It contains training, development, and test data with equally many positive and negative examples of strings from 1,800 regular languages across 16 different classes. The data is provided in small (1k), medium (10k), and large (100k) sizes with the smaller preparations fully contained in the each of the larger.

MLRegTest is designed to investigate a number of factors which may contribute to the learning performance. These factors are shown in Table 1, along with the values considered in this work. A regular language from the data set is uniquely identified by the factors

Factor Name	Description (Levels)
`Alph`	Alphabet size (4, 16)
`Tier`	(3, 4, 7, 16)
`Class`	Language class (SL, coSL, TSL, TcoSL, SP, coSP, LT, TLT, PT, LTT, TLTT, PLT, TPLT, SF, Zp, Reg)
`k`	Factor width (0, 2, 3, 4, 5, 6)
`j`	Threshold size (0, 1, 2, 3, 5)
`i`	Index (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
`TrainSize`	Size of training set (Small: 1k, Mid: 10k, Large: 100k)
`TestType`	Type of test set: whether strings are (S)hort or (L)ong, (R)andom or (A)dversarial (SR, SA, LR, LA)
`AlgoType`	State merging algorithm (RPNI, EDSM, ALERGIA)

Table 1: Factors comprising the experimental design.

alphabet size (`Alph`), tier size (`Tier`), language class (`Class`), factor width (`k`), threshold size (`j`), and index (`i`).

The sixteen language classes in MLRegTest are organized according to descriptive complexity using logic and model-theoretic signatures of strings. Five kinds of logic are considered: Monadic Second Order (MSO), First Order (FO), Propositional (Prop), Conjunctions of Negative literals (CNL) and Disjunctions of Positive Literals (DPL). CNL and DPL are complements; both are fragments of Prop, which is properly contained in FO, which is a fragment of MSO. Three basic model-theoretic signatures are considered: the successor model ($\lhd$), the tier-successor model ($\lhd_T$), and the precedence model ($<$), which at lower logical levels organize languages according to the concepts of substring, tier-substring, and subsequence, respectively. A tier T is a subset the alphabet Σ; they are the salient symbols. Non-tier elements are neutral and can be freely inserted or deleted (Lambert, 2023).

Choice of logic and model-theoretic signature define a logical language, whose formulas define formal languages. At the highest level, MSO, these converge to the regular languages (Reg): $\text{MSO}(\lhd) = \text{MSO}(\lhd_T)$ for all $T = \text{MSO}(<)$. MLRegTest considers an additional subclass of purely periodic languages they call Zp.[3] At the FO level, the classes of languages defined by the logical languages are nested: $\text{FO}(\lhd) \subsetneq \text{FO}(\lhd_T)$ for all $T \subsetneq \text{FO}(<)$. At the propositional level we have $\text{Prop}(\lhd)$ is properly included by $\text{Prop}(\lhd_T)$, but both are incomparable with $\text{Prop}(<)$. An analgous relationship holds for CNL and DPL: if X$\in${CNL, DPL}, it holds that $\text{X}(\lhd)$ is properly included by $\text{X}(\lhd_T)$, but both are incomparable with. $\text{X}(<)$. Many of these classes are well-studied and go by the names shown in Table 2. Overviews of these classes and their relationships are discussed by McNaughton and Papert (1971); Thomas (1982, 1997); Rogers and Pullum (2011); Rogers et al. (2013); Rogers and Lambert (2019). Lambert et al. (2021) discusses aspects of many of these classes learnability.

Some of these are are known by other names. For example, what is called coSP here is also known as the $1/2$ level of the Straubing Hierarchy (Straubing, 1985; Pin and Weil, 1997). To this list of thirteen classes, MLRegTest also includes Piecewise Locally Testable (PLT)

3. Examples include languages like $\{w : |w|_a \mod 2 = 0\}$, i.e. the language whose words contain an even number of a symbols.

Logic	Representation	Language Name	Abbrevation
MSO	$\lhd$ $\lhd_T$ $<$	Regular	Reg
FO	$\lhd$ $\lhd_T$ $<$	Locally Threshold Testable Tier Locally Threshold Testable Star-Free	LTT TLTT SF
Prop	$\lhd$ $\lhd_T$ $<$	Locally Testable Tier Locally Testable Piecewise Testable	LT TLT PT
CNL	$\lhd$ $\lhd_T$ $<$	Strictly Local Tier Strictly Local Strictly Piecewise	SL TSL SP
DPL	$\lhd$ $\lhd_T$ $<$	Complement of Strictly Local Complement of Tier Strictly Local Complement of Strictly Piecewise	coSL coTSL coSP

Table 2: Logical characterizations and names of classes in MLRegTest.

langauges given by Prop$(<, \lhd)$ and Tier Piecewise Locally Testable (TPLT) langauges given by Prop$(<, \lhd_T)$. While both of these classes are contained within FO$(<)$, it is the case that Prop$(<, \lhd)$ properly contains $(\lhd)$, and Prop$(<, \lhd_T)$ properly contains $(\lhd_T)$. van der Poel et al. (2023) verify that the languages representing a class A in MLRegTest belong to A and do not belong to any class B, where B is a subclass of A or is incomparable with A. We refer the reader to van der Poel et al. (2023) for additional details regarding these parameters.

MLRegTest contains four variations of testing data based on the length of the strings generated from the language (short or long) and the method of generation (random or adversarial). The short testing data contains equally many strings of all lengths between 20 and 29 whereas the long shifts this to be between 31 and 50. Similarly, the random test sets were sampled randomly without replacement while the adversarial test sets pair each positive example with a corresponding negative example such that the edit distance is one.

Our experiments are not conducted on the entirety of MLRegTest. In particular, we excluded languages with an alphabet of size 64. This is because it was discovered that the data sets of those languages were confounded by an error likely introduced by unicode normalization. Therefore the total number of languages in our study is 1,140 with every class represented.

3.3. Evaluation

As advocated by Demšar (2006) and Stąpor (2018) and employed by van der Poel et al. (2023), we utilize non-parametric methods in our analysis. This minimizes the assumptions made about our data, as no distribution is required, while still allowing us to assign rankings to the variables under question. We report aggregate accuracy for the models as the primary

measurement of performance. A model in our experiments is specified by the choice of regular language, size of the training data, and algorithm for building the automata. As a result, we aimed to generate 1,140 (regular languages) $\times$ 3 (training data sizes) = 3,420 models for each algorithm, each of which would be evaluated on 4 test sets, yielding 13,680 observations per algorithm. We achieved this goal with EDSM and RPNI, but only obtained 11,136 observations for ALERGIA because it was time-intensive.

These observations are then aggregated over the variables `Tier`, `k`, `j`, `i` and `Alph`. The analysis presented in §4 holds these as blocking variables, while considering factors such as `Class`, `TrainSize`, `TestType`, and `AlgoType` as treatment variables for which to compute average accuracy over. The Friedman rank sum test is used to determine if the factor being held for treatment contributes significantly to predicting model accuracy by holding the null hypothesis that all variations of the value contribute equally. We conduct additional analysis using the Nemenyi-Wilcoxon-Wilcox all-pairs test to identify exactly which pairs differ significantly.

Because EDSM, RPNI, and ALERGIA return finite state machines, these can be compared with the minimal deterministic finite state acceptors for the languages in MLRegTest. There are many forms this comparison can take. We chose to compare the sizes of the finite state machines as a first step. In future work, this analysis can be refined, examining other details such as the subset relationships of the corresponding languages.

4. Results

We follow a similar analysis to van der Poel et al. (2023) so that their results on neural networks can be directly compared to the performance of the state-merging algorithms presented here.

4.1. Training Size

As the training data is nested such that increasing the size strictly increases the data seen, we expect the models trained on the larger splits to improve in accuracy.

Small	**Mid**	**Large**
0.585	0.624	0.655

Table 3: Average accuracy by `TrainSize`.

As can be seen in Table 3, the accuracy of the models does indeed increase on average with the size of the training data. The Friedman rank sum test also finds that varying the size of the training data leads to statistically significant differences in model performance (Friedman chi-squared = 542.58, df = 2, p-value $< 2.2 \times 10^{-16}$). The Nemenyi-Wilcoxon-Wilcox test similarly rejects the null hypothesis with p-value at or below $< 2.9 \times 10^{-14}$ for all pairs. This is a good sanity check to ensure the integrity of the experimental setup and analysis pipeline.

Nonetheless, it was surprising that even on the Large training set the mean accuracy was not particularly high. It is puzzling that these algorithms do not perform as well we might expect, even when trained on the large test sets. After all, it has been proven that RPNI will

identify the target language in the limit provided the input data includes a characteristic sample. EDSM uses heuristics to guide its search, so no theoretical results exits, but analysis of ALERGIA does provide similar learnability results (see also de la Higuera and Thollard (2000); Clark and Thollard (2004)). The answer must be the large data sets do not include a characteristic sample. One reason may be that the shortest strings in the training sets in MLRegTest are at least of length 20. If the characteristic samples for the regular languages in MLRegTest necessarily contains strings of shorter lengths, then these algorithms will not achieve exact identification. Therefore, follow-up experiments which included shorter strings are reported in Section 5. The results of these experiments are encouraging.

4.2. Test Set Type

We turn our attention to the effect of `TestType` on model accuracy. Table 4 shows the average accuracy scores for each preparation of test data.

LA	SA	LR	SR
0.534	0.539	0.696	0.718

Table 4: Average accuracy by `TestType`.

Visually, the differences in scores look significant and the Friedman rank sum test confirms this suspicion by rejecting the null hypothesis that performance across test types is the same (Friedman chi-squared = 782.96, df = 3, p-value < 2.2e-16). Additional analysis using the Nemenyi-Wilcoxon-Wilcox all-pairs test finds that all four preparations of test data differ significantly from each other at or below 1.9e-09. These findings are consistent with the results on neural network architectures presented by van der Poel et al. (2023).

Figure 2: Accuracy by `Class` and `TestType`.

Abbrev.	Logical Level	Lang. Classes
CNL	Conjunction of Negative Literals	SL, SP, TSL
DPL	Disjunction of Positive Literals	coSL, coSP, TcoSL
PROP	Propositional	LT, PLT, PT, TLT, TPLT
FO	First Order	LTT, TLTT, SF
REG	Monadic First Order	Zp, Reg

Table 5: Language classes grouped by logical level.

4.3. Language Class

In this section we investigate the impact that `Class` has on model accuracy. To do this, we consider two groupings of the 16 language classes based on the weakest logic capable of describing the language (CNL, DPL, PROP, FO, REG) and the order relation used to represent strings from that language (successor, precedence, tier-successor).

4.3.1. Logical Level

The language classes grouped by their logical level are shown in Table 5. If we find a significant difference in accuracy based on the logical level, one might expect accuracy to decrease as we ascend to more expressive logical levels.

REG	PROP	DPL	CNL	FO
0.595	0.620	0.628	0.630	0.649

Table 6: Average accuracy by logical level.

The Friedman test finds that logical level is a significant factor in predicting accuracy (Friedman chi-squared = 25.133, df = 4, p-value = 4.73e-05).

	CNL	DPL	FO	PROP
DPL	0.510	–	–	–
FO	0.774	0.048	–	–
PROP	0.084	0.878	0.002	–
REG	0.016	0.544	$1.9e-4$	0.978

Table 7: Nemenyi-Wilcoxon-Wilcox all-pairs test results comparing by logical level.

Interrogating the Nemenyi-Wilcoxon-Wilcox all-pairs test tells a different story from predictions based on logical complexity. While REG is generally the hardest to learn, there are few significant differences anywhere. The only ones which appear significant are properly FO languages from properly propositional ones and properly regular ones. This differs from the analysis by van der Poel et al. (2023) on neural network architectures which found everything FO and below to be indistinguishable.

4.3.2. Order Relation

Next we consider the language classes grouped by order relation as shown by Table 8. From the results above we might expect OTHER to obtain the lowest accuracy, as it contains the regular class, followed by the rest.

Abbrev.	Order Relation	Lang. Classes
SUCC	Successor	SL, coSL, LT, LTT
PREC	Precedence	coSP, PT, SF, SP
TSUCC	Tier Successor	TcoSL, TLT, TLTT, TSL
OTHER	Other	LP, TLP, Reg, Zp

Table 8: Langauge classes grouped by order relation.

Consulting the Friedman rank sum test finds order relation to be a significant factor (Friedman chi-squared = 13.95, df = 3, p-value = 0.002974). Further examining the p-values from the Nemenyi-Wilcoxon-Wilcox all-pairs test, see Table 10, finds significant differences between successor and tier-successor languages.

OTHER	SUCC	PREC	TSUCC
0.616	0.622	0.633	0.635

Table 9: Average accuracy by successor relation.

	OTHER	PREC	SUCC
PREC	0.569	–	–
SUCC	0.041	0.527	–
TSUCC	0.836	0.144	0.003

Table 10: Nemenyi-Wilcoxon-Wilcox all-pairs test results comparing by successor relation.

4.4. Analysis by Machine Sizes

We now turn away from the parametric statistical tests to qualitatively inspect the sizes of the learned machines relative to the source machines used for generating the data sets. When state-merging algorithms return a machine M with more states than the target machine, it means not enough states were merged, and therefore M makes more distinctions than necessary (a kind of overfitting). On the other hand, if M is smaller than the target, it means too few distinctions were made and M overgeneralizes. On average the learned FAs are larger than the data source FAs in roughly $^3/_4$ experiments, smaller in about $^1/_4$, and exactly the same size in the remaining 2.5% of cases. The appendix includes figures which give more details on this analysis.

For EDSM and RPNI the learned machines are much larger, averaging state counts that are 59.71 (std=163.18, median=6.63) and 91.66 (std=213.32, median=13.15) times larger,

respectively. While ALERGIA also learns machines that are larger than the source, they average a relatively modest 3.28 (std=6.32, median=1.00) times increase in size. As expected, larger training data sets resulted in models with larger numbers of states (see Figure 4 in the appendix). These size increases were especially notable in the Zp languages where EDSM and RPNI averaged size factors of 231.11 and 311.44 each. Meanwhile, machines trained on SP and coSP languages had mean factors (ALERGIA=1.38, EDSM=18.20, RPNI=33.04) which were among the lowest. This wide range in variability results in the large standard deviations seen.

We also investigated the relationship between learned machine size and accuracy. This revealed that, across all test types, larger machines were less accurate for EDSM and RPNI. However, this trend was reversed for ALERGIA which saw larger machines perform better (see Figure 5 in the appendix). When the machine sizes exactly matched the source, the accuracy was higher, but not perfect. In these cases the average accuracies were 0.89 (std=0.19, median=1.00) for EDSM, 0.94 (std=0.15, median=1.00) for RPNI, and 0.56 (std=0.10, median=0.50) for ALERGIA. It is curious that ALERGIA sees essentially no improvement in this case. In fact, despite the median state count of the learned machine exactly matching that of the source machine, we see in the next section that this does not translate to better overall generalizations for ALERGIA.

4.5. State-Merging versus Neural Networks

Finally, we compare the accuracy of the three state merging algorithms both with each other, as well as with the neural network architectures studied by van der Poel et al. (2023). The accuracy results are shown in Table 11. In order to make an apples-to-apples comparison, the accuracies for the neural networks were recalculated from the raw experimental observations they collected after removing all observations pertaining to languages with an alphabet size of 64.[4]

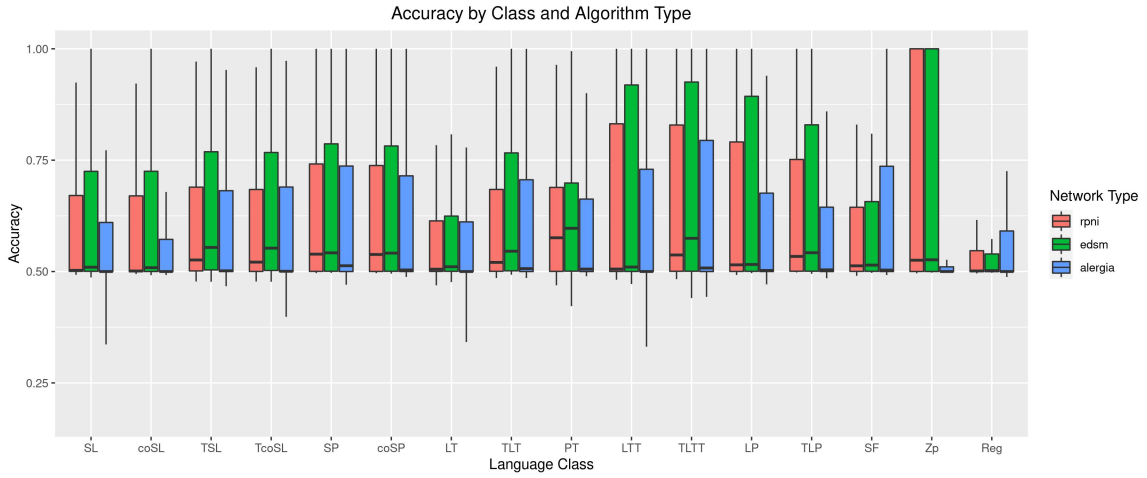

Figure 3: Accuracy by `Class` and `AlgoType`.

4. These observations are located in the analysis folder in the public repository here: https://github.com/heinz-jeffrey/subregular-learning.

Generally, the state-merging algorithms tend to perform worse than the neural network approaches across the board.

	Small	Mid	Large
RPNI	0.589	0.626	0.650
EDSM	0.607	0.638	0.655
ALERGIA	0.560	0.609	0.660
Simple RNN	0.708	0.751	0.797
GRU	0.668	0.738	0.837
LSTM	0.688	0.774	0.852
2-layer LSTM	0.682	0.770	0.864
Transformer	0.637	0.704	0.749

Table 11: Average accuracy by `AlgoType` and `TrainSize` along side the results on neural networks found by van der Poel et al. (2023).

The Friedman rank sum test finds, like for neural network architectures, that the state-merging algorithm employed contributes to significant differences in accuracy (Friedman chi-squared = 169.27, df = 2, p-value $< 2.2 \times 10^{-16}$). For all train set sizes, the Nemenyi-Wilcoxon-Wilcox all-pairs test finds the state-merging algorithms significantly different. For the RPNI/ALERGIA comparison, the p-value is 0.001; for EDSM/ALERGIA it is less than 2×10^{-16}; and for EDSM/RPNI it equals 3.7×10^{-14}. The results for all three sizes of training data are shown in Table 12. A summary of these findings is shown in Figure 3.

	EDSM	RPNI	EDSM	RPNI	EDSM	RPNI
RPNI	3.7×10^{-14}	–	1.5×10^{-6}	–	2.4×10^{-4}	–
ALERGIA	$< 2 \times 10^{-16}$	0.001	2.4×10^{-10}	0.310	0.002	0.840
	Small		**Mid**		**Large**	

Table 12: Nemenyi-Wilcoxon-Wilcox all-pairs test results by `AlgoType` and `TrainSize`.

5. Adding Short Strings to Training

As follow-up experiments, we created roughly 1,000 short training strings for the 1,140 languages used in this study, and then added these to the Small, Mid, and Large training sets to create training sets we call Small$^+$, Mid$^+$, and Large$^+$, respectively. We then trained EDSM, RPNI, and ALERGIA on these new training sets and evaluated them on each of the test sets. The results, presented in Table 13, show that the short strings dramatically improve accuracy, which we discuss in more detail further below.

Short training strings were constructed as follows. For each language L, 25 positive and 25 negative strings of length n were created for all n between 1 and 19 inclusive. Recall that the shortest string in MLRegTest was of length 20. The positive (negative) strings were sampled with replacement by intersecting the acceptor for L (for the complement of

	RPNI	EDSM	ALERGIA
Small	0.589	0.607	0.560
Small$^+$	0.744	0.756	0.578
Mid	0.626	0.638	0.609
Mid$^+$	0.772	0.784	0.622
Large	0.650	0.655	0.660
Large$^+$	0.796	0.811	0.637

Table 13: Average accuracy by `AlgoType` and training regiment.

L) with an acceptor for Σ^n and randomly selecting paths assuming a uniform probability over transitions at each state. This was the same method used by MLRegTest.

In some cases, there are no positive (or negative) strings of a certain length, and in such a case no strings were drawn. Therefore, not all languages have the same number of strings in the short training data, nor do they necessarily have the same number of positive and negative strings. The minimum number of strings in the short training samples was 675, the maximum number was 950, and the median was 875. The mean was 872.7 with a standard deviation of 69.2. The minimum proportion of positive strings in the short training samples was 29.6%, the maximum proportion was 70.3%, and the median was 50%. The mean was 49.0% with a standard deviation of 6.79%.

The effects of these short training strings for EDSM and RPNI are striking. There is about a 15 point increase in accuracy for these algorithms by including the shorter training strings. In particular, for both EDSM and RPNI, the average accuracy of models trained on Small$^+$ is about 10 points higher than the average accuracy of the models trained on the Large training sample. To put this in perspective, this means that models trained on just less than 2,000 strings outperformed models trained on 100,000 strings. Clearly, for EDSM and RPNI, it is data quality and not data quantity that makes a difference. This supports the hypothesis that characteristic samples for these languages contain shorter strings, whose omission from the original MLRegTest helps explain why these state-merging algorithms do not perform so well.

For ALERGIA, the short training strings help, but only by about 2 points in the Small and Mid comparisons. Comparing average accuracy on Large and Large$^+$, the short training strings actually hamper its performance. In this regard, it is worth noting that ALERGIA sees the most benefit from training sample size, as compared to EDSM and RPNI. The average accuracy for ALERGIA increases by about 10 points from the Small to Large training samples, while for EDSM and RPNI it only increases by about 5-6 points. This suggests that for ALERGIA, data quantity may be more important than data quality, which may follow from the statistical methods used in ALERGIA. It would be interesting to see whether ALERGIA's performance improves by sampling many more shorter strings.

6. Conclusions

Our experiments allow us to conclude with confidence that, among the state-merging algorithms investigated, EDSM performs best in terms of accuracy on the learning of regular languages. The accuracies for RPNI follow closely behind, while ALERGIA performs considerably worse, doing only a few percent better than random on the adversarial test sets.

Overall, these algorithms don't seem to keep pace with the neural network architectures on the original training samples in MLRegTest. None of the state-mergers average above 70% accuracy on the Large training sets while neural approaches generally exceed 80%.

However, it is very interesting to observe how the performance of the state-merging algorithms improved with the inclusion of short training data, even on the Small training data set. An immediate goal of future research is to run the neural network models on the Short$^+$, Mid$^+$, and Large$^+$ training data sets and to make this comparison. If it is the case that the state-merging algorithms outperform the neural networks on Small$^+$ data, that would indicate that grammatical inference has an important role to play in generalizing well in the absence of big data.

Additional avenues for future work include developing a statistical comparison which takes into account the relative simplicity of the state-merging algorithms when compared to the neural network architectures. Using the number of statistical parameters as a proxy for complexity might be a place to start. While the state-merging algorithms have a limited number of parameters, there are some which could be varied to possibly improve performance. For ALERGIA in particular, the accuracy may be significantly improved by tuning the assignment of final states with the, currently unused, development sets.

Acknowledgments

Data set creation, model training, and evaluation were completed on the Stony Brook SeaWulf HPC cluster maintained by Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University and made possible by NSF grant #1531492.

We also thank Sicco Verweer for invaluable assistance with FlexFringe as well as the ICGI reviewers for their feedback which improved this work.

Appendix A. Additional Figures Analyzing the Sizes of Learned Models

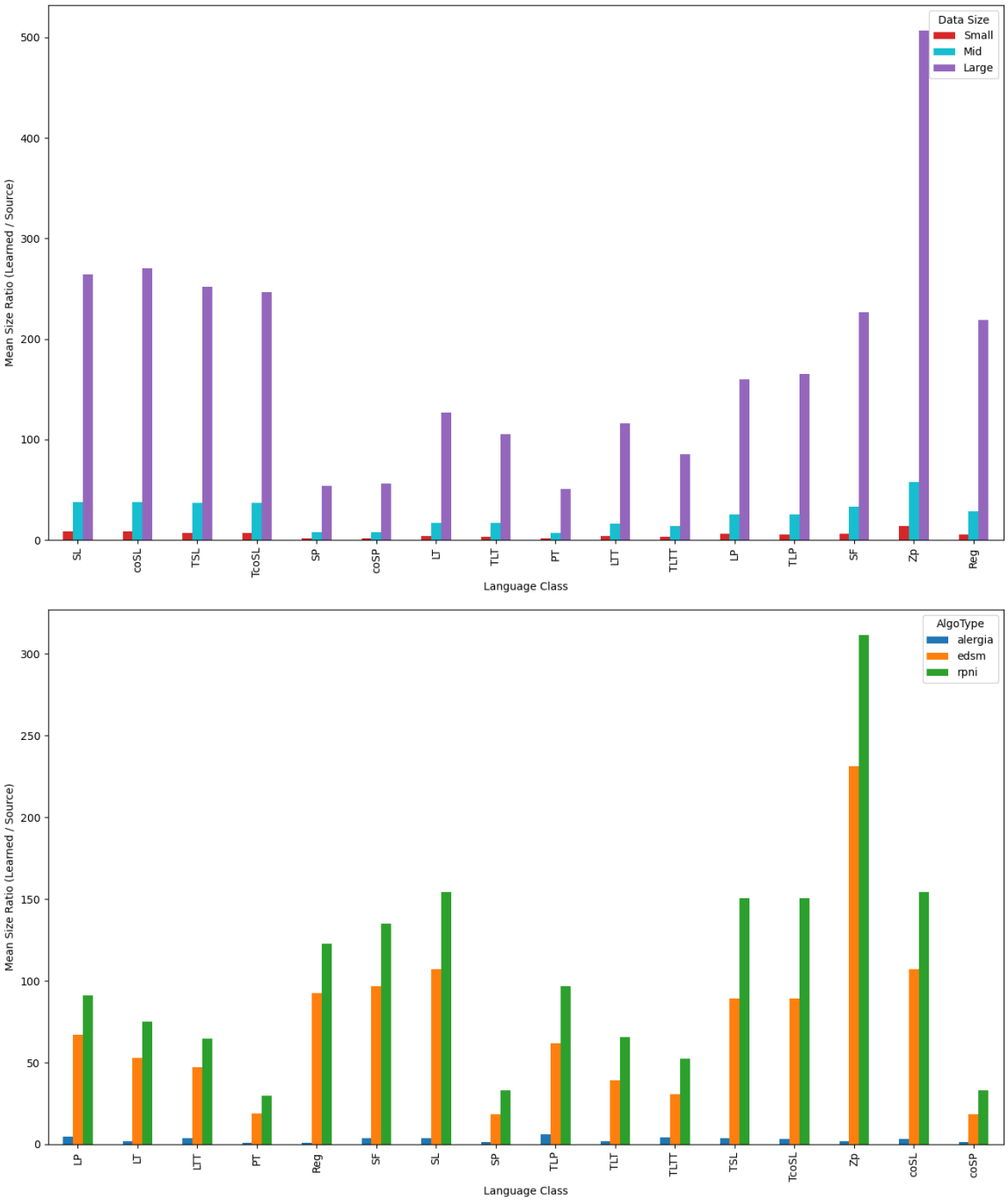

Figure 4: (top) Machine size by training data and `Class`. (bottom) Machine size by `AlgoType` and `Class`.

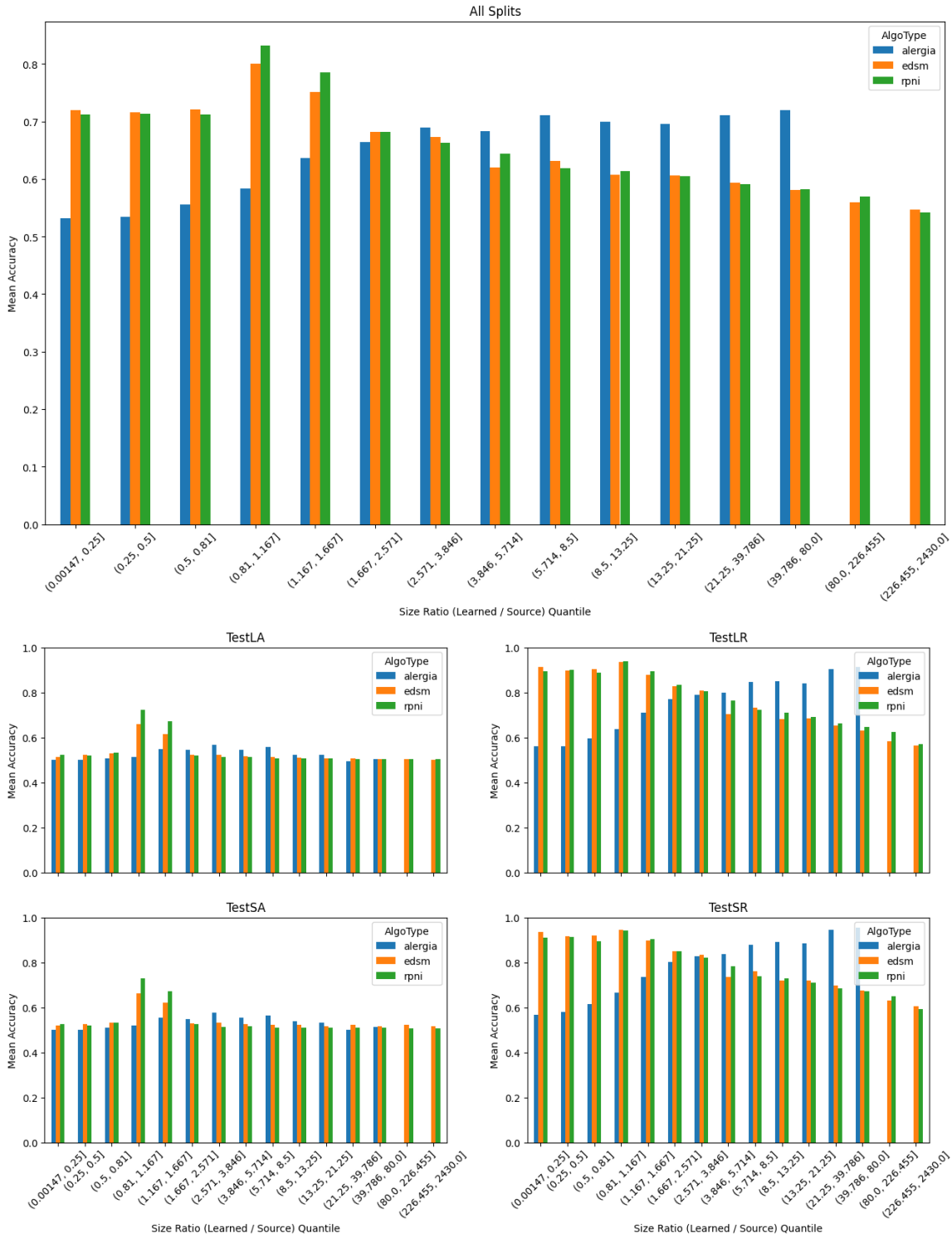

Figure 5: Accuracy by machine size, test split, and `AlgoType`.

References

R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO (Theoretical Informatics and Applications)*, 33(1): 1–20, 1999.

Rafael C. Carrasco and José Oncina. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21-23, 1994, Proceedings*, pages 139–152, 1994.

Alexander Clark and Franck Thollard. Pac-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497, 2004.

Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

Colin de la Higuera and Franck Thollard. Identification in the limit with probability one of stochastic deterministic finite automata. In A.L. de Oliveira, editor, *Grammatical Inference: Algorithms and Applications, Proceedings of ICGI '00*, volume 1891 of *Lecture Notes in Computer Science*, pages 15–24. Springer-Verlag, 2000.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, dec 2006. ISSN 1532-4435.

Jeffrey Heinz, Colin de la Higuera, and Menno van Zaanen. *Grammatical Inference for Computational Linguistics*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, 2015.

Dakotah Lambert. Relativized adjacency. *Journal of Logic Language and Information*, 2023. doi: https://doi.org/10.1007/s10849-023-09398-x.

Dakotah Lambert, Jonathan Rawski, and Jeffrey Heinz. Typology emerges from simplicity in representations and learning. *Journal of Language Modelling*, 9(1):151–194, 2021.

Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. In Vasant Honavar and Giora Slutzki, editors, *Grammatical Inference*, pages 1–12, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, 1971.

José Oncina and Pedro García. Inferring regular languages in polynomial updated time. In *Pattern Recognition and Image Analysis*. World Scientific, 1992. URL https://doi.org/10.1142/9789812797902_0004.

J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30(4):383–422, 1997. doi: https://doi.org/10.1007/BF02679467.

James Rogers and Dakotah Lambert. Some classes of sets of structures definable without quantifiers. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 63–77, Toronto, Canada, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5706. URL https://www.aclweb.org/anthology/W19-5706.

James Rogers and Geoffrey Pullum. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342, 2011.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. Cognitive and sub-regular complexity. In Glyn Morrill and Mark-Jan Nederhof, editors, *Formal Grammar*, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

Katarzyna Stąpor. Evaluating and comparing classifiers: Review, some recommendations and limitations. In Marek Kurzynski, Michal Wozniak, and Robert Burduk, editors, *Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017*, pages 12–21, Cham, 2018. Springer International Publishing.

Howard Straubing. Finite semigroup varieties of the form V * D. *Journal of Pure and Applied Algebra*, 36:53–94, 1985. ISSN 0022-4049. doi: https://doi.org/10.1016/0022-4049(85)90062-3.

Ole Tange. *GNU Parallel 2018*. Ole Tange, March 2018. ISBN 9781387509881. doi: 10.5281/zenodo.1146014. URL https://doi.org/10.5281/zenodo.1146014.

Wolfgang Thomas. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences*, 25:370–376, 1982.

Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, chapter 7. Springer, 1997.

Sam van der Poel, Dakotah Lambert, Kalina Kostyszyn, Tiantian Gao, Rahul Verma, Derek Andersen, Joanne Chau, Emily Peterson, Cody St. Clair, Paul Fodor, Chihiro Shibata, and Jeffrey Heinz. Mlregtest: A benchmark for the machine learning of regular languages, 2023. URL https://arxiv.org/abs/2304.07687.

Sicco Verwer and Christian Hammerschmidt. Flexfringe: Modeling software behavior by learning probabilistic automata, 2022. URL https://arxiv.org/abs/2203.16331.

Sicco Verwer and Christian A. Hammerschmidt. flexfringe: A passive automaton learning package. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 638–642, 2017. doi: 10.1109/ICSME.2017.58.