Extracting Weighted Finite Automata from RNNs via iterative partitioning and spectral learning

Sandamali Yashodhara Wickramasinghe[1,*], Jacob M. Howe[1] and Laure Daviaud[2]

[1]City St George's, University of London, Northampton Square, London EC1V 0HB, UK
[2]University of East Anglia, Research Park, Norwich NR4 7TJ, UK

Abstract

This paper proposes an algorithm to extract a Weighted Finite Automaton (WFA) from a Recurrent Neural Network (RNN), by adapting the L^* algorithm in combination with spectral learning using Singular Value Decomposition (SVD). The method introduces a discrete abstraction of the RNN's hidden state space to enable approximate equivalence queries within the L^* framework. Through these queries, counterexamples are identified and used to iteratively refine the prefix and suffix sets that define the Hankel matrix, thereby improving the accuracy of successive WFA hypotheses. This work lays the groundwork for integrating discrete clustering with spectral learning in the context of WFA extraction from RNNs, an area that remains underexplored, with potential applications in sequence modelling and analysis. Future work will focus on implementing and empirically validating the proposed approach.

Keywords

Recurrent Neural Networks, Weighted Finite Automata, Explainable AI, Interpretability

1. Introduction

This research explores the extraction of Weighted Finite Automata from Recurrent Neural Networks. Recurrent Neural Networks (RNNs) are often referred to as black boxes due to their lack of interpretability. Understanding the behaviour of RNNs is important to enhance their interpretability and stability, and this understanding can be achieved by extracting symbolic models such as automata. Most research on extracting knowledge from RNNs has focused on Deterministic Finite Automata (DFA).

DFA extraction is suitable for RNNs trained as binary classifiers (producing Boolean outputs). In contrast, for RNNs with real-valued outputs, such as probabilities, extracting Weighted Finite Automata is more appropriate [1].

One approach to extract automata from RNNs is through compositional techniques, where a WFA is derived by dividing the RNN state space into clusters and identifying the transitions and weight matrices to construct the automaton [2, 3]. Another approach involves exact learning methods, such as the L^* algorithm [4], where an automaton is extracted by posing queries about the language learned by the RNN. Angluin [4] introduced this L^* algorithm in 1987 for inferring a DFA from an unknown target system, which was later extended to extract DFAs from RNNs [5]. In the L^* algorithm, two types of queries are employed in the extraction process: membership queries and equivalence queries. A membership query asks whether a specific string belongs to the target language, while an equivalence query checks whether the language recognised by the extracted automaton is equivalent to that of the target.

The work in [5] introduced a novel approach for extracting a DFA from an RNN using the L^* algorithm and demonstrated strong performance in accurately representing the language learned by the RNN. Later on, the work was extended to extract PDFAs (Probabilistic Deterministic Finite Automata) from RNNs [6]. Even though the L^* algorithm was designed to extract a DFA, later on, [7] proposed a

OVERLAY 2025, 7th International Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, October 26th, Bologna, Italy
*Corresponding author.
sandamali.wickramasinghe@citystgeorges.ac.uk (S. Y. Wickramasinghe); j.m.howe@citystgeorges.ac.uk (J. M. Howe); l.daviaud@uea.ac.uk (L. Daviaud)
0000-0002-5550-1940 (S. Y. Wickramasinghe); 0000-0001-8013-6941 (J. M. Howe); 0000-0002-9220-7118 (L. Daviaud)


variation of it for WFA extraction, namely the spectral learning algorithm [8]. Building upon this, [9] conducted a study where a WFA is extracted without requiring access to the internal mechanisms of the RNN or the original training data. The work presented in [10] also used spectral learning to extract WFAs from second-order RNNs.

The use of exact learning methods for extracting WFAs from RNNs remains relatively underexplored. While membership queries are straightforward in this setting, the main challenge lies in performing equivalence queries, as a formal equivalence between a WFA and an RNN is undecidable [11].

In [5], this was addressed by using an abstraction of the RNN's hidden state space while doing the equivalence queries. By clustering similar hidden states into discrete regions, a finite-state abstraction can be constructed and compared against the behaviour of the extracted automaton. This method enables an approximate but feasible equivalence check. The same abstraction-based approach was later extended to WFA extraction in the study presented by [1].

Our proposed contributions are twofold: first, we aim to integrate spectral learning with Singular Value Decomposition (SVD) to perform membership queries in the extraction of a WFA from an RNN using the L^* algorithm; second, we propose addressing the challenge of equivalence queries by iteratively constructing a discrete abstraction of the RNN state space to enable effective comparison. This framework builds upon the foundational work of [8] and extends the ideas explored in [5] toward the extraction of WFAs.

In contrast to prior approaches such as [9, 12], which construct WFAs using spectral learning over fixed sets of prefixes and suffixes, we propose an iterative spectral learning framework. This approach incrementally expands the set of prefixes and suffixes based on the learning process, allowing for dynamic refinement until equivalence is achieved, potentially improving both adaptability and efficiency.

Additionally, we propose incorporating a discrete abstraction network during equivalence queries by incrementally partitioning the RNN hidden state space. This builds upon the preliminary ideas in [5] and can be further refined using advanced partitioning strategies, including hyperplane-based segmentation [13], k-means clustering [2], and the K-DCP method introduced in [3].

While the use of the weighted L^* algorithm for WFA extraction has been explored in [1], its integration with spectral learning [8] with a discrete state abstraction remains under-explored. Furthermore, the iterative construction of an abstraction network using the aforementioned partitioning techniques has not been applied in this context. Our proposed framework aims to bridge these gaps and provide a new direction for extracting WFAs from RNNs.

2. Related work

The association between automata and RNNs has been studied in the context of both binary classification problems [14, 5] and quantitative tasks [9, 10, 6, 1, 3, 2, 12]. The earliest works, such as [15, 16], laid the foundation for understanding the relationship between automata and RNNs. In 1996, Omlin and Giles [17] demonstrated that the state space of an RNN trained on a regular grammar tends to cluster in a way that resembles the structure of a finite automaton. Subsequent studies continued to explore this line of research, including [18, 14], further validating the connection between RNN dynamics and automata representations.

The work presented in [5] is considered state-of-the-art in DFA extraction and employs the L^* algorithm, an exact learning approach. This method was extended to quantitative RNNs in a probabilistic setting in [6], where probabilistic deterministic finite automata (PDFAs) were extracted. Subsequently, [1] further advanced this line of research by extracting WFAs using a regression-based method to perform equivalence checks.

In [7], a weighted variation of the L^* algorithm was introduced specifically for WFA extraction with the use of spectral learning for WFA introduced in [8]. Unlike the original L^* algorithm by [4], which constructs an observation table through membership queries to a teacher (i.e., the target system), this variation replaces the observation table with a Hankel matrix populated by real-valued entries. A WFA

is then derived from this matrix using spectral learning techniques, with rank factorisation and Singular Value Decomposition (SVD) being core components of the process. Building on [8], [10] demonstrated that second-order RNNs with linear activation functions are natural extensions of, and computationally equivalent to, WFAs, establishing a fundamental connection between the two models.

The works in [9] and [12] focus on extracting WFAs from RNNs producing real-valued outputs over sequential data without accessing internal states of the black-box models. These studies adopt the spectral learning framework introduced in [8], treating the RNN as an oracle and analysing its outputs to infer the automaton. However, they do not incorporate equivalence queries in the learning process.

A compositional approach is presented in [3], which diverges from the exact learning approach. Their Decision Guided Extraction algorithm leverages the target RNN's decision-making process and contextual information to guide the extraction. This method has shown efficiency in larger-scale tasks, addressing scalability challenges in approximating RNN behaviour. Similarly, [2] employed a compositional strategy using k-means clustering to extract WFAs from RNNs trained on natural language data.

While exact learning methods offer strong accuracy guarantees, they often face scalability and computational complexity limitations. In contrast, compositional approaches, though potentially less precise, provide more scalable solutions. Therefore, combining these methods could yield a more effective framework for automata extraction from RNNs, enhancing both accuracy and scalability [1, 6, 5].

3. Background

This section introduces the technical notation that will be used throughout the paper. A finite alphabet (a finite set of elements) Σ is fixed, and the set of all finite strings (i.e. a finite sequence of elements) w over Σ is denoted by Σ^*. The empty string of length 0 is denoted by ε. The length of a string $w \in \Sigma^*$ is denoted by $|w|$. Given two strings u and v, the concatenation of the two strings is $w = uv$. In that case, u is called a prefix of w and v a suffix of w. A set $\mathcal{P}$ of strings is said to be prefix closed if for all strings $w \in \mathcal{P}$, all the prefixes of w are also in $\mathcal{P}$.

Definition 1. Hankel matrix [8]: *Let f be a function $f : \Sigma^* \to \mathbb{R}$. The Hankel matrix of f is a bi-infinite matrix $\mathcal{H} \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$, whose entries are defined as $\mathcal{H}(u, v) = f(uv)$ for any $u, v \in \Sigma^*$.*

Rows of a Hankel matrix can thus be seen as indexed by prefixes and columns by suffixes. Only the finite sub-blocks of $\mathcal{H}$ are of interest, and a sub-block can be defined using a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, where $\mathcal{P}, \mathcal{S} \subseteq \Sigma^*$. The sub-block of $\mathcal{H}$ defined by $\mathcal{B}$ is the sub-matrix of $\mathcal{H}_\mathcal{B}$ restricted to the rows indexed by $\mathcal{P}$, and columns indexed by $\mathcal{S}$. We will mainly be interested in basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ that are *prefix closed* that is to say $\mathcal{P}, \mathcal{S} \subseteq \Sigma^*$ are prefix-closed and *complete*, meaning that the rank of the sub-block $\mathcal{H}_\mathcal{B}$ equals the rank of $\mathcal{H}$.

Definition 2. Weighted Finite Automaton (WFA): *A WFA A is a tuple $\langle \Sigma, Q, \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma \in \Sigma} \rangle$, where Σ is the finite alphabet, Q is the finite set of states, α_0 and α_∞ are vectors of size $|Q|$ called the initial and final vectors, and A_σ is the transition matrix corresponding to $\sigma \in \Sigma$ with size $|Q| \times |Q|$ and real-valued entries. A WFA also can be represented linearly as a triplet $\langle \alpha_0, (A_\sigma)_{\sigma \in \Sigma}, \alpha_\infty \rangle$ [8, 12].*

The weight of a string $w = \sigma_1 \sigma_2 \dots \sigma_n$ in a WFA A is computed as $A(w) = \alpha_0^\top A_w \alpha_\infty = \alpha_0^\top A_{\sigma_1} A_{\sigma_2} \dots A_{\sigma_n} \alpha_\infty$.

Definition 3. Recurrent Neural Network (RNN): *A Recurrent Neural Network is defined as a tuple $R = (h_0, g_R, f_R)$, where $h_0 \in \mathbb{R}^d$ is the initial hidden state with $d \in \mathbb{N}$, $g_R : \mathbb{R}^d \times \Sigma \to \mathbb{R}^d$ is the transition function, and $f_R : \mathbb{R}^d \to \mathbb{R}$ is the output function mapping a hidden state to a real value. For an input string $w \in \Sigma^*$, the RNN R produces a real-valued output, defining a function $f_R : \Sigma^* \to \mathbb{R}$ that maps strings to real values.*

4. Extracting WFAs from RNNs

This section describes the proposed algorithm for extracting a WFA from an RNN using the L^* algorithm (see Algorithm 1).

4.1. L^* Algorithm for WFA learning from an RNN

Angluin's L^* algorithm learns a minimal automaton using a *minimally adequate teacher*, a theoretical construct capable of answering queries about an unknown target system. The algorithm relies on two types of queries: *membership queries* and *equivalence queries*. In the context of learning a WFA from a target function $f : \Sigma^* \to \mathbb{R}$, the learner interacts with the teacher as follows [7]:

In *membership queries*, the learner queries the teacher for the value $f(w)$ of a specific string w, and records the result. Based on the collected information, the learner constructs a hypothesised WFA A. *Equivalence queries* are then used to verify whether the hypothesised WFA A correctly computes the target function f. If it does not, the teacher provides a *counterexample*– a string on which the behaviour of A differs from f (i.e. $A(w) \neq f(w)$). The learner uses this counterexample to refine the hypothesis.

In this setting, the RNN is a model that produces real-valued outputs and serves as the teacher, meaning that both membership and equivalence queries are directed to the RNN. The responses to membership queries are recorded in a *Hankel matrix* (see Definition 1), which can then be used to derive a minimal WFA through SVD-based rank factorisation (as described in Section 4.2).

4.2. Hankel matrix to WFA

Theorem 1. *Let $f : \Sigma^* \to \mathbb{R}$ be a function over Σ^*. Then, f is computable by a WFA if and only if the rank of its Hankel matrix is finite. In that case, this rank is equal to the minimum number of states of any WFA computing f [19, 20].*

As shown in [8], a WFA can be constructed from the Hankel matrix $\mathcal{H}$ of a function $f : \Sigma^* \to \mathbb{R}$ using a sub-block $\mathcal{H}_\mathcal{B} \in \mathbb{R}^{p \times s}$, defined by a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ with $|\mathcal{P}| = p$ prefixes and $|\mathcal{S}| = s$ suffixes, where $\mathcal{P}$ is prefix-closed and $\mathcal{B}$ spans the row and column spaces of $\mathcal{H}$ (see Algorithm 1 lines 2–5).

For a sub-block $\mathcal{H}_\mathcal{B}$ of rank r, a minimal WFA is computed via SVD, where $\mathcal{H}_\mathcal{B} = U\Lambda V^\top$, with $U \in \mathbb{R}^{p \times r}$ and $V \in \mathbb{R}^{s \times r}$ orthogonal, and $\Lambda \in \mathbb{R}^{r \times r}$ a diagonal matrix containing singular values of $\mathcal{H}_\mathcal{B}$, yielding a rank factorization $\mathcal{H}_\mathcal{B} = PS$ with $P = U\Lambda \in \mathbb{R}^{p \times r}$ and $S = V^\top \in \mathbb{R}^{r \times s}$.

Additionally, let $\mathcal{H}_\sigma \in \mathbb{R}^{p \times s}$ be the sub-block defined by $\mathcal{H}_\sigma(u, v) = \mathcal{H}(u\sigma, v)$ for $u \in \mathcal{P}$, $v \in \mathcal{S}$, and $\sigma \in \Sigma$, let $h_{\mathcal{P},\epsilon} \in \mathbb{R}^p$ be the vector with coordinates $h_{\mathcal{P},\epsilon}(u) = \mathcal{H}(u, \epsilon)$, and let $h_{\epsilon,\mathcal{S}} \in \mathbb{R}^s$ be the vector with coordinates $h_{\epsilon,\mathcal{S}}(v) = \mathcal{H}(\epsilon, v)$. The WFA $A = \langle \alpha_0, (A_\sigma)_{\sigma\in\Sigma}, \alpha_\infty \rangle$ is then defined with initial weights $\alpha_0^\top = h_{\epsilon,\mathcal{S}}^\top S^+$, final weights $\alpha_\infty = P^+ h_{\mathcal{P},\epsilon}$ and transition matrices $A_\sigma = P^+ \mathcal{H}_\sigma S^+$, where P^+ and S^+ is the Moore–Penrose pseudo inverse of P and S respectively [8].

The prefixes $\mathcal{P}$ and suffixes $\mathcal{S}$ required to construct the basis can be derived from strings generated over the alphabet on which the RNN was trained. These sets can be generated in lexicographical order, expanding systematically, or sampled from a uniform distribution over the alphabet [12].

4.3. Equivalence queries and the abstraction of the state space

When using the L^* algorithm for extracting a WFA from an RNN, a key challenge is performing equivalence queries due to the differing natures of RNNs (continuous state space in $\mathbb{R}^d$) and WFAs (finite discrete states), which complicates establishing exact equivalence. To address this, a discrete abstraction A_P of the RNN's hidden state space can be used to compare its behaviour with the WFA A during the extraction process.

Definition 4. *Suppose A_P is a hypothesised abstraction of the RNN state space. Given an RNN R with state space $\mathbb{R}^d$, a partitioning function $P : \mathbb{R}^d \to Q_{A_P}$ maps each hidden state to a discrete state in the set $Q_{A_P} = \{q_1, q_2, \ldots, q_{|Q_{A_P}|}\}$, where $|Q_{A_P}|$ is the number of states (clusters) in the abstraction A_P. The*

Algorithm 1 WFA Extraction from RNN via Hankel Matrix and State Abstraction

Require: $R, \Sigma, \mathcal{B} = (\mathcal{P}, \mathcal{S}), A_P$
Ensure: WFA $A = \langle \alpha_0, \{A_\sigma\}_{\sigma \in \Sigma}, \alpha_\infty \rangle$

```
1: while equivalence not achieved do
2:   if P is not prefix-closed then
3:     P ← P ∪ {u' | u' is a prefix of u ∈ P}
4:   end if
5:   Construct WFA A = (α_0, (A_σ)_{σ∈Σ}, α_∞)
6:   for each test string w ∈ Σ* do
7:     if A(w) ≠ A_P(w) and A(w) ≠ f_R(w) then
8:       counterexample ← w and add prefixes and suffixes of w to P or S
9:       continue outer loop
10:    else if A(w) ≠ A_P(w) and A_P(w) ≠ f_R(w) then
11:      Refine abstraction A_P
12:    end if
13:  end for
14: end while
15:
16: return WFA A
```

abstraction A_P represents a partitioning of the RNN state space, where each state $q_i \in Q_{A_P}$ corresponds to a cluster of hidden states.

Equivalence queries perform two tasks: (1) return a counterexample string $w \in \Sigma^*$ if the WFA A is incorrect (i.e., $A(w) \neq f_R(w)$), and add its prefixes to $\mathcal{P}$ and suffixes to $\mathcal{S}$ to refine the basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ (see Algorithm 1, lines 7–8); (2) refine the abstraction A_P by partitioning $\mathbb{R}^d$ into finer clusters (see Algorithm 1, lines 10–11).

To construct the abstraction A_P, the RNN's state space $\mathbb{R}^d$ can be iteratively partitioned into clusters during equivalence queries (see Definition 4). Initially, the entire state space is treated as a single cluster. If $A_P(w) \neq f_R(w)$, for a string w, while performing equivalence queries, the abstraction A_P should be refined, and clustering methods can be applied.

The refinement of the abstraction A_P could be realised using these compositional methods, such as decision-guided extraction using stepwise predictions [3] and k-means based clustering of hidden states [2] or hyperplane-based partitioning [13] during equivalence queries, where the hidden state space is iteratively divided into partitions. This iterative refinement of A_P continues until the WFA A accurately approximates the RNN's behaviour, with A_P grouping similar hidden states and separating dissimilar ones to facilitate effective comparison and identify counterexamples for refining $\mathcal{B}$.

The termination of the algorithm is intended to be guided by an error tolerance parameter, which specifies the permissible discrepancy between the RNN and the extracted WFA during equivalence queries. This parameter influences the granularity of the resulting WFA: a lower tolerance is expected to yield a finer approximation, while a higher tolerance produces a coarser model. In theory, termination occurs when no further counterexamples are identified, indicating agreement between the WFA and the RNN on the tested strings, or when resource limits, such as a predefined time limit [13], are reached, since convergence is inherently approximate. Because WFAs and RNNs are fundamentally different constructs, exact equivalence cannot be established. In practice, therefore, a test set can be used to evaluate the extracted WFA, providing an empirical measure of how closely it approximates the behaviour of the RNN under the chosen tolerance.

5. Conclusion

This paper explores an algorithm to extract a Weighted Finite Automaton (WFA) from a Recurrent Neural Network (RNN) with real-valued outputs, using a variation of the L^* algorithm. The approach combines spectral learning with Singular Value Decomposition (SVD) to handle membership queries and introduces a discrete abstraction of the RNN's hidden state space to approximate equivalence queries. The goal is to investigate whether such a method can yield an accurate approximation of the RNN.

A key factor influencing the quality of the extracted WFA is the dynamic selection of prefixes and suffixes used to build the Hankel matrix. Counterexamples generated during equivalence queries are added to the basis, enabling iterative refinement that improves the accuracy of each successive WFA hypothesis. Consequently, the algorithm does not require access to the full training dataset; only the alphabet of the language for which the RNN was trained is needed, along with access to the RNN hidden state space, making the process more efficient.

To the best of our knowledge, the integration of spectral learning with the weighted L^* algorithm using a discrete abstraction for equivalence queries to extract WFAs has not been previously explored. This work provides a foundation for interpretable WFA extraction from RNNs, with potential applications in sequence modelling and analysis. In future, we aim to extend the implementation and theoretical framework beyond the proposed work.

Acknowledgments. Work supported by the EPSRC grant EP/T018313/1.

References

[1] T. Okudono, M. Waga, T. Sekiyama, I. Hasuo, Weighted Automata Extraction from Recurrent Neural Networks via Regression on State Spaces, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 5306–5314. doi:10.1609/aaai.v34i04.5977.

[2] Z. Wei, X. Zhang, M. Sun, Extracting Weighted Finite Automata from Recurrent Neural Networks for Natural Languages, 2022, pp. 370–385. doi:10.1007/978-3-031-17244-1_22.

[3] X. Zhang, X. Du, X. Xie, L. Ma, Y. Liu, M. Sun, Decision-Guided Weighted Automata Extraction from Recurrent Neural Networks, Proceedings of the AAAI Conference on Artificial Intelligence 35 (2021) 11699–11707. doi:10.1609/aaai.v35i13.17391.

[4] D. Angluin, Learning regular sets from queries and counterexamples, Information and Computation 75 (1987) 87–106. doi:10.1016/0890-5401(87)90052-6.

[5] G. Weiss, Y. Goldberg, E. Yahav, Extracting automata from recurrent neural networks using queries and counterexamples (extended version), Machine Learning 113 (2024) 2877–2919. doi:10.1007/s10994-022-06163-2.

[6] G. Weiss, Y. Goldberg, E. Yahav, Learning Deterministic Weighted Automata with Queries and Counterexamples, Advances in Neural Information Processing Systems 32 (2019).

[7] B. Balle, M. Mohri, Learning Weighted Automata, in: International Conference on Algebraic Informatics, Springer International Publishing, 2015, pp. 1–21. URL: https://link.springer.com/10.1007/978-3-319-23021-4_1. doi:10.1007/978-3-319-23021-4_1.

[8] B. Balle, X. Carreras, F. M. Luque, A. Quattoni, Spectral learning of weighted automata, Machine Learning 96 (2014) 33–63. doi:10.1007/s10994-013-5416-x.

[9] S. Ayache, R. Eyraud, N. Goudian, Explaining Black Boxes on Sequential Data using Weighted Automata, International Conference on Grammatical Inference 93 (2018) 81–103. URL: http://arxiv.org/abs/1810.05741.

[10] G. Rabusseau, T. Li, D. Precup, Connecting Weighted Automata and Recurrent Neural Networks through Spectral Learning, 2019. doi:doi.org/10.48550/arXiv.1807.01406.

[11] R. Marzouk, C. de la Higuera, Distance and equivalence between finite state machines and recurrent

neural networks: Computational results, arXiv preprint arXiv:2004.00478 (2020). doi:10.48550/arXiv.2004.00478.

[12] R. Eyraud, S. Ayache, Distillation of weighted automata from recurrent neural networks using a spectral approach, Machine Learning 113 (2024) 3233–3266. doi:10.1007/s10994-021-05948-1.

[13] S. Y. Wickramasinghe, J. M. Howe, L. Daviaud, Extracting Deterministic Finite Automata from RNNs via Hyperplane Partitioning and Learning, in: Proceedings of the 17th International Joint Conference on Computational Intelligence, 2025. To appear.

[14] Q. Wang, K. Zhang, A. G. Ororbia II, X. Xing, X. Liu, C. L. Giles, An Empirical Evaluation of Rule Extraction from Recurrent Neural Networks, Neural Computation 30 (2018) 2568–2591. doi:10.1162/neco_a_01111.

[15] A. Cleeremans, D. Servan-Schreiber, J. L. McClelland, Finite State Automata and Simple Recurrent Networks, Neural Computation 1 (1989) 372–381. doi:10.1162/neco.1989.1.3.372.

[16] C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, Y. C. Lee, Learning and Extracting Finite State Automata with Second-Order Recurrent Neural Networks, Neural Computation 4 (1992) 393–405. doi:10.1162/neco.1992.4.3.393.

[17] C. W. Omlin, C. Giles, Extraction of rules from discrete-time recurrent neural networks, Neural Networks 9 (1996) 41–52. doi:10.1016/0893-6080(95)00086-0.

[18] H. Jacobsson, Rule Extraction from Recurrent Neural Networks: A Taxonomy and Review, Neural Computation 17 (2005) 1223–1263. doi:10.1162/0899766053630350.

[19] J. W. Carlyle, A. Paz, Realizations by stochastic finite automata, 1971.

[20] M. Fliess, Matrices de Hankel, Journal de Mathématiques Pures et Appliquées 53 (1974).