Rule Extraction from Recurrent Neural Networks: A Taxonomy and Review

Henrik Jacobsson
henrikj@ida.his.se

Abstract

Rule extraction (RE) from recurrent neural networks (RNNs) refers to finding models of the underlying RNN, typically in the form of finite state machines, that mimic the network to a satisfactory degree. RE from RNNs can be argued to allow a deeper and more profound form of analysis of RNNs than other, more or less *ad hoc* methods. RE may give us understanding of RNNs in the intermediate levels between quite abstract theoretical knowledge of RNNs as a class of computing devices and quantitative performance evaluations of RNN instantiations. The development of techniques for extraction of rules from RNNs has been an active field since the early nineties. In this paper, the progress of this development is reviewed and analysed in detail. In order to structure the survey and to evaluate the techniques, a taxonomy, specifically designed for this purpose, has been developed. Moreover, important open research issues are identified, that, if addressed properly, possibly can give the field a significant push forward.

Technical report: HS-IKI-TR-04-002
School of Humanities and Informatics
University of Skövde
Box 408
SE-541 28 Skövde, Sweden

Contents

Rule Extraction from Recurrent Neural Networks: A Taxonomy and Review

Henrik Jacobsson

Abstract

Rule extraction (RE) from recurrent neural networks (RNNs) refers to finding models of the underlying RNN, typically in the form of finite state machines, that mimic the network to a satisfactory degree. RE from RNNs can be argued to allow a deeper and more profound form of analysis of RNNs than other, more or less *ad hoc* methods. RE may give us understanding of RNNs in the intermediate levels between quite abstract theoretical knowledge of RNNs as a class of computing devices and quantitative performance evaluations of RNN instantiations. The development of techniques for extraction of rules from RNNs has been an active field since the early nineties. In this paper, the progress of this development is reviewed and analysed in detail. In order to structure the survey and to evaluate the techniques, a taxonomy, specifically designed for this purpose, has been developed. Moreover, important open research issues are identified, that, if addressed properly, possibly can give the field a significant push forward.

1 Introduction

In this paper, techniques for extracting rules (or finite state machines) from discrete-time recurrent neural networks (DTRNNS, or simply RNNs) are reviewed. We propose a new taxonomy for classifying existing techniques, present the techniques, evaluate them, and produce a list of open research issues that need to be addressed.

By rule extraction from RNNs (hereafter denoted RNN-RE) we refer to the process of finding/building (preferably comprehensible) formal models/machines that mimic the RNN to a satisfactory degree. The connection between RNNs and formal models of computation is almost as old as the study of RNNs themselves as the origins of these fields are largely overlapping. The study of neural networks once coincided with the study of computation in the binary recurrent network implementations of finite state automata of the theoretical work on nervous systems by McCulloch & Pitts (1943) (an interesting overview of this topic is found in Forcada, 2002.) This common heritage has been flavouring the development of the digital computer (von Neumann 1956, Minsky 1967) although our current computer systems are very far from being models of the nervous system.

In the early nineties, the research on recurrent neural networks was revived. When Elman introduced his, quite well known, simple recurrent network (SRN) (Elman 1990), the connection between finite state machines and neural networks

was again there from the start. In his paper, the internal activations of the networks were compared to the states of a finite state machine.

1.1 Topic delimitation

Since the early nineties, an abundance of papers has been written on recurrent neural networks[1], and many of them have dealt explicitly with the connection between RNNs and state machines. Many contributions have been theoretical, establishing the connection between (analogue) RNNs (or other dynamical systems) and traditional (discrete) computational devices (Crutchfield & Young 1990, Servan-Schreiber, Cleeremans & McClelland 1991, Crutchfield 1994, Kolen 1994, Horne & Hush 1994, Siegelmann & Sontag 1995, Casey 1996, Tiňo, Horne, Giles & Collingwood 1998, Blank & 24 co-authors 1999, Omlin & Giles 2000, Sima & Orponen 2003, Hammer & Tiňo 2003, Tiňo & Hammer 2003). These papers cover a wide spectrum of highly interesting and important theoretical insights, but in this paper we will not dwell on these theoretical issues. First of all because it is not the focus of this survey, but also because some of these papers are already much like surveys themselves, summarising earlier findings.

On the pragmatic side we find papers describing techniques for transforming state machines into RNNs (rule insertion) and/or for transforming RNNs into state machines (rule extraction) (Omlin & Giles 1992, Giles & Omlin 1993, Das, Giles & Sun 1993, Alquézar & Sanfeliu 1994*a*, Omlin & Giles 1996*a*, Omlin & Giles 1996*c*, Omlin, Thornber & Giles 1998, Omlin & Giles 2000, Carrasco, Forcada, Muñoz & Neco 2000, Carrasco & Forcada 2001). This paper, however, deals exclusively with algorithms for performing *rule extraction* from RNNs.

This paper is also not about analysis tools of RNNs other than just RE. There are a multitude of methods used to analyse RNNs, and a survey on this issue should definitely be written as well. A brief (and most probably inconclusive) list of examples of other analysis tools that have been used on RNNs is: Hinton diagrams (e.g. Hinton 1990, Niklasson & Bodén 1997), Hierarchical Cluster Analysis (e.g. Cleeremans, McClelland & Servan-Schreiber 1989, Elman 1990, Servan-Schreiber, Cleeremans & McClelland 1989, Sharkey & Jackson 1995, Bullinaria 1997), simple state space plots (e.g. Giles & Omlin 1993, Zeng, Goodman & Smyth 1993, Gori, Maggini & Soda 1994, Niklasson & Bodén 1997, Tonkes, Blair & Wiles 1998, Tonkes & Wiles 1999, Rodriguez, Wiles & Elman 1999, Rodriguez 1999, Tabor & Tanenhaus 1999, Linåker & Jacobsson 2001), activation values plotted over time (e.g. Husbands, Harvey & Cliff 1995, Meeden 1996, Ziemke & Thieme 2002), iterated maps (e.g. Wiles & Elman 1995), vector flow fields (e.g. Rodriguez et al. 1999, Rodriguez 1999), external behaviour analysis of RNN-controlled autonomous robotic controllers (e.g. Husbands et al. 1995, Meeden 1996), weight space analysis (e.g. Bodén, Wiles, Tonkes & Blair 1999, Tonkes & Wiles 1999), dynamical systems theory (e.g. Tonkes et al. 1998, Rodriguez et al. 1999, Rodriguez 1999, Boden, Jacobsson & Ziemke 2000), and ordinary quantitative evaluations of RNN performance for different domains (basically every single paper where an RNN is applied).

Unlike previous surveys on rule extraction (Andrews, Diederich & Tickle 1995, Tickle, Andrews, Golea & Diederich 1997, Tickle & Andrews 1998), this

[1] Many of these are summarised in Kremer (2001) and Barreto, Araújo & Kremer (2003).

paper deals exclusively with rule extraction from *recurrent* neural networks (resulting in quite different evaluation criteria than in previous RE surveys, as you will see in Section 4). In fact, many of the RE approaches for non-recurrent networks could potentially be used on RNNs, or at least on non-recurrent networks in temporal domains (e.g. Craven & Shavlik 1996, Sun, Peterson & Sessions 2001). There are also other symbolic learning techniques for "training" finite automata on symbolic sequence domains directly, without taking the extra step of training a neural network, that could be mentioned (Sun & Giles 2001, Cicchello & Kremer 2003). These techniques are certainly interesting in themselves and should also be compared to RNN-RE techniques experimentally, but this is something which is not further examined in this paper.

So, to summarize, this paper is oriented solely around RNN-RE techniques, but this field is closely related to the above mentioned areas. It may also be worth to mention that, as a review of techniques, this paper is not a tutorial. So, for readers interested in replicating the techniques, we refer to the cited papers.

1.2 Contents overview

First, in Section 2 we will discuss how rule extraction fits into a broader context of RNN research. In Section 3 we will describe RNNs, finite state machines, and describe common characteristics of RNN-RE algorithms. The evaluation criteria underlying the construction of a taxonomy for appropriately classifying and describing RNN-RE algorithms is described in Section 4. The techniques are then described in Section 5 and some important criticisms of RNN-RE in general is discussed in Section 6. The set of existing RNN-RE techniques are then discussed in light of the evaluation criteria in Section 7 and open research issues are summarised in Section 8. In Section 9, we present some conclusions.

2 Some thoughts on recurrent neural networks analysis

In theory, RNNs are Turing equivalent[2] (Siegelmann & Sontag 1995), and can therefore compute whatever function any digital computer can compute. But we also know that to get the RNN to perform the desired computations is very difficult (Bengio, Simard & Frasconi 1994). This leaves us in a form of knowledge vacuum; we know that RNNs *can be* immensely powerful computational devices, and we also know that finding the correct instantiations of RNNs that perform these computations could very well be an insurmountable obstacle, but we do not have the means for efficiently determining the computational abilities of our current RNN instantiations[3]. On a less theoretical level, we can simply evaluate the performance of different RNNs to see to which extent we solved

[2] Actually McCulloch and Pitts (McCulloch & Pitts 1943) determined this equivalence already in 1943, for discrete networks (Medler 1998).

[3] We do know some things of course, for example, that for an RNN successfully trained on a regular language such that it *robustly* models the language, there exists a finite state machine, equivalent to the RNN (Casey 1996). The problem is, however, to know whether a network is robustly modelling a regular grammar before we attempt to extract this grammar from the network.

the intrinsic learning problem for a specific domain. Such studies are conducted in virtually all papers applying RNNs on a domain, and in some cases more systematic studies are presented (Miller & Giles 1993, Horne & Giles 1995, Alquézar, Sanfeliu & Sainz 1997). But even something as simple as evaluating the performance of an RNN on a specific domain has some intrinsic problems since implicit aspects of the evaluation procedure can have a significant impact on the estimated quantitative performance (Jacobsson & Ziemke 2003*a*). So, what we need to do is in-depth analyses of RNN instantiations to uncover the actual behaviour of RNN instantiations. But is the stage set for conducting such studies?

If we compare the study of artificial neural networks in general (ANNs, i.e. not RNNs only) with the study of the human brain, which after all is the main source of inspiration in connectionism (Medler 1998), there are of course some quite obvious differences. The complexity of the brain by far surpasses even the most ambitious neural network implementations. But there are some reasons for why the study of ANNs still can help us in the study of the brain; first of all, that we can create artificial models of the brain to an, in principle, arbitrary degree of complexity. But more importantly, because the ANN allows us to do things that are not possible or ethical to do with a human brain. ANNs are kind enough to let us (among other things):

- reproduce results with arbitrarily high accuracy,
- repeat experiments without much additional effort after the framework for the first experiment has been implemented, giving us the theoretical ability to get ever more precise results,
- duplicate networks (and distribute them among research colleagues),
- study the effect of damage to the network under controlled conditions,
- study the interrelation of evolutionary and life-time learning (e.g. instincts),
- do nonperturbative studies of internal properties to an arbitrary degree of detail.

In other words, ANNs are almost perfect experimental subjects. Very few scientific communities have the luxury of studying entities with properties so inviting for conducting research on them.

The crux is, however, that the ANNs are not always easy to understand. The framework for understanding them is certainly there, as the list above clearly demonstrates, but the obstacle is the *complexity* of the models themselves. An ANN adapted to solve a specific task is, by its domain-driven emergent behaviour and its holistic representation of knowledge, a complex entity to study. This is especially true for recurrent neural networks, where the networks, due to their recursive structure, can be complex dynamical systems with chaotic behaviours. Actually, the analysis problems may lead to the use of too simplistic models, e.g. smaller networks and toy problem domains, just to be able to analyse (or visualise) the results. One may wonder how many published networks with just two or three state nodes (or hidden nodes) had their specific topology chosen just to make the plotting of their internal activations possible.

So, in summary, although we know the theoretical possibilities and limitations of RNNs, we still cannot be sure how these theories manifest themselves in practice. Although we can do almost whatever we want to the networks, we do need *tools* to analyse and understand them. Casey (1996) stated, about previous (non-RE) RNN analysis approaches, that "none of these measures is sufficient for understanding RNN behavior in general" (p. 1136) and indicated that rule (or finite state machine) extraction may be one way to alleviate this. So, perhaps one approach towards doing "nonperturbative studies of internal properties to an arbitrary degree of detail" on RNNs (i.e. the last point of the list above), is to extract finite state machines from them, hopefully providing us with a clearer view of the internal properties of instantiations of RNNs. Rule extraction may turn out to be an essential ingredient for conducting methodical testing of instantiations of RNN solutions, focusing on analysis on the level between deriving abstract theoretical possibilities of RNNs as a class of computational devices and on the other hand simplistic quantitative performance estimations of RNN instantiations.

3 Background

An RNN processes sequences of data (input) and generates a responses (output) in a discrete time manner. The RNN processes information by using its internal continuous state space as an implicit, holistic, memory of past input patterns (Elman 1990). In the extraction of rules from an RNN, the continuous state space is approximated by a finite set of states and the dynamics of the RNN is mapped to transitions among this discrete set of states.

We will now give a brief definition of what constitutes a recurrent neural network in the scope of this paper. A brief introduction to finite state machines (FSMs) will also be given, since the extracted rules are typically represented as such. A more detailed description of what RNN-RE algorithms typically constitute will then follow.

3.1 Recurrent neural networks

To give a detailed review of the achievements in RNN research and the vast variety of different RNN architectures is far beyond the scope of this paper. Instead, a set of identified common features of most RNN architectures will be described at abstract enough a level to hopefully incorporate most networks to which the existing RNN-RE algorithms could be applied and also abstract enough to see the striking similarities of RNN computation with the computation in finite state machines (see Section 3.2). Readers with no prior experience of RNNs can find more detailed description and well developed classifications of RNNs in Kolen & Kremer (2001), Kremer (2001) or Barreto et al. (2003).

Only a few of the many RNN architectures have at all been used in the context of rule extraction, e.g. simple recurrent networks (SRNs, Elman, 1990) and more commonly second-order networks (e.g. Sequential Cascaded Networks, SCNs, Pollack, 1987). These models differ somewhat in their functionality and how they are trained. But the *functional dependencies* are, at some level of abstraction, basically the same, which is going to be exploited in the definition below.

A *Recurrent Neural Network* R is a 6-tuple $R = \langle I, O, S, \delta, \gamma, \mathbf{s}^0 \rangle$ where, $I \subseteq \mathbb{R}^{n_i}$ is a *set of input vectors*, $S \subseteq \mathbb{R}^{n_s}$ is a *set of state vectors*, $O \subseteq \mathbb{R}^{n_o}$ is a *set of output vectors*, $\delta : S \times I \to S$ is the *state transition function*, $\gamma : S \times I \to O$ is the *state interpretation function*, and $\mathbf{s}^0 \in S$ is the *initial state vector*. $n_i, n_s, n_o \in \mathbb{N}$ are the *dimensionalities* of the input, state and output spaces respectively.

Often (or perhaps always) the input, state and output are restricted to *hypercubes* with all elements limited to real numbers (or, of course, rational approximations of real numbers when simulated) between zero and one or minus one and one. When training the networks, the two functions δ and γ are typically adjusted to produce the desired output according to some training set. For a sequence of input vectors $(\mathbf{i}^0, \mathbf{i}^1, \mathbf{i}^2, \ldots, \mathbf{i}^\ell)$ the state is updated according to $\mathbf{s}^t = \delta(\mathbf{s}^{t-1}, \mathbf{i}^t)$ and the output according to $\mathbf{o}^t = \gamma(\mathbf{i}^t, \mathbf{s}^{t-1})$. The functional dependencies are depicted in Figure 1.

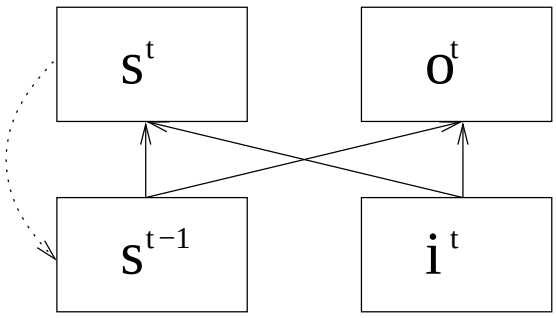

Figure 1: The functional dependencies of the input, state and output of an RNN.

Note that the weights, biases, activation functions and other concepts we typically associate with neural networks are all hidden in the state transition function δ and state interpretation function γ. The reason for this is that, as far as RNN-RE algorithms are concerned, the fact that the networks have adaptive weights and can be trained, is of less importance. An interesting consequence of the abstract nature of this RNN description, and that this is all that is required to continue describing RNN-RE algorithms, is that it tells something about the portability of the algorithms (cf. section 4.2). There are simply not many assumptions and requirements of the underlying RNNs, which means that they are portable to more RNN types than they would be otherwise. There are a few assumptions though, e.g. that states should cluster in the state space as a result of the training (Cleeremans et al. 1989, Servan-Schreiber et al. 1989). Some, more implicit, assumptions are also the target for some of the criticism of RE from RNNs (Kolen 1993, Kolen 1994), which will be discussed later in Section 6 (more implicit assumptions are discussed in Section 7.5.3).

If we consider a Simple Recurrent Network (Elman 1990) as one example of an RNN, the functional dependencies are a specific case of the ones defined above, see figure 2. The output $\mathbf{o}^t$ of an SRN is solely determined by the state $\mathbf{s}^t$ which in turn depends on $\mathbf{s}^{t-1}$ and $\mathbf{i}^t$. It may seem counter-intuitive that the SRN has the same functional dependencies as the general description of RNNs

above, but by rules of transitivity, it does. It can, however, be quite fruitful to consider this more specific functional graph when extracting rules as no more than necessary dependencies need to be considered in the extraction process.

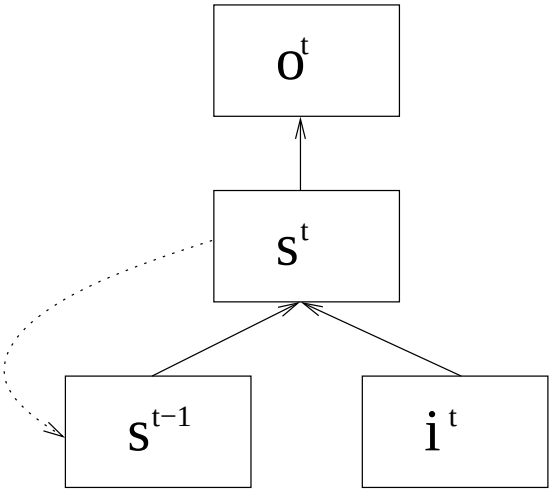

Figure 2: The functional dependencies of the input, state and output of an SRN. The functional dependencies are, by transitivity, equivalent to that of the general RNN description in Figure 1.

3.2 Finite state machines

The rules extracted from RNNs are almost exclusively represented as finite state machines (FSMs). The description here will be kept brief and for a full discussion of what a regular language is and what other classes of languages there are, interested readers are referred to Hopcroft & Ullman (1979).

A *Deterministic Mealy Machine* M is a 6-tuple $M = \langle X, Y, Q, \delta, \gamma, q^0 \rangle$ where, X is the finite *input alphabet*, Y is the finite *output alphabet*, Q is a *finite set of states*, $\delta : Q \times X \rightarrow Q$ is the *transition function*, $\gamma : Q \times X \rightarrow Y$ is the *output function*, and $q^0 \in Q$ is the *initial state* (note the similarities with the definition of RNNs in Section 3.1).

In cases where the output alphabet is binary the machine is often referred to as a *finite state automata* (FSA). In an FSA, the output is interpreted as an *accept* or *reject* decision determining whether an input sequence is accepted as a grammatical string or not.

There are actually two different models for how an FSM can be described; *Mealy* (as above) or *Moore* machines that, although they are quite different from each other, are computationally equivalent (Hopcroft & Ullman 1979). Moore machines generate outputs based only on the current state and Mealy machines on the transitions between states, i.e. the output function, γ, is for a Moore machine $\gamma : Q \rightarrow Y$ and for a Mealy machine $\gamma : Q \times X \rightarrow Y$.

In *deterministic* machines, an input symbol may only trigger a single transition from one state to *exactly one* state (as in the definition above). In a

nondeterministic machine, however, a state may have zero, one or more outgoing transitions triggered by the same input, i.e. the transition function, δ, is $\delta : Q \times X \rightarrow 2^Q$ (a function to the power set of Q) instead of $\delta : Q \times X \rightarrow Q$. That means that in a nondeterministic machine, a symbol may trigger one or more transitions from a state, or even no transition at all (since $\emptyset \in 2^Q$). We will denote nondeterministic machines *incomplete* if at least one $q \in Q$ and $x \in X$ such that $\delta(X, Q) = \emptyset$. Deterministic and nondeterministic machines are computationally equivalent, although nondeterministic machines can typically be much more compact (i.e. have less states) than their deterministic counterpart. Deterministic finite state machines and deterministic finite state automata, will be abbreviated DFM and DFA respectively.

If the outgoing transitions of the states in a nondeterministic machine, $\delta : Q \times X \rightarrow 2^Q$, are also associated with probability distributions over 2^Q, the machine will be denoted as a *stochastic* FSM[4].

In summary, there are four types of FSMs; deterministic Moore machine, deterministic Mealy machine, nondeterministic Moore machine, and nondeterministic Mealy machine, see Figure 3 for examples. Moreover, the nondeterministic machines can be stochastic as well if transition probabilities are also encoded.

For a more detailed description of deterministic and nondeterministic, Mealy and Moore machines, proof of equivalence and a "standard" minimisation algorithm, see Hopcroft & Ullman (1979).

3.3 The basic recipe for RNN rule extraction

The algorithms described in this paper have many features in common as enlisted in Table 1.

1. Quantisation of the continuous state space of the RNN, resulting in a discrete set of states.
2. State and output generation (and observation) by feeding the RNN input patterns.
3. Rule construction based on the observed state transitions.
4. Rule set minimisation.

Table 1: The common "ingredients" of RNN-RE algorithms.

The continuous state space of the RNN needs to be mapped into a finite set of discrete states corresponding to the states of the resulting machine. We will refer to the states of the network as *microstates* and the finite set of quantised states of the network as *macrostates*. The macrostates are basically what the RE algorithm "sees" of the underlying RNN, whereas the actual state of the network, the microstates, are hidden. The act of transforming the microstates into macrostates is a critical part of RNN-RE algorithms (ingredient one in

[4]Denoted *probabilistic automata* by Rabin (1963).

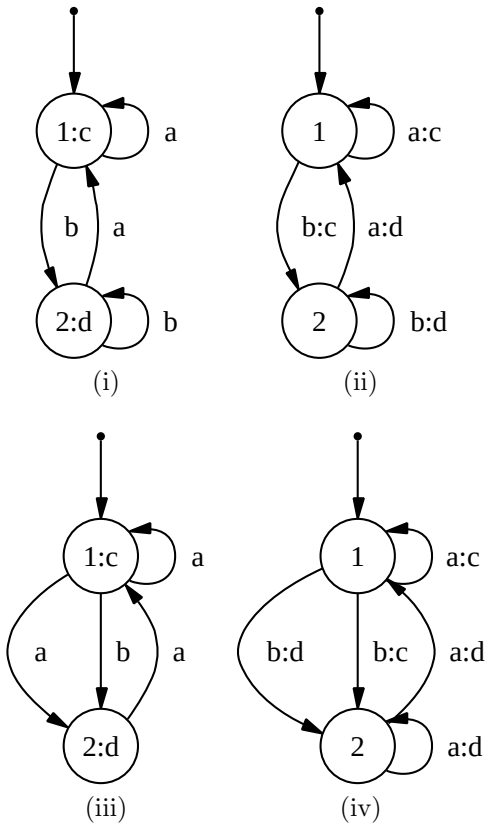

Figure 3: Examples of (non-equivalent) different finite state machine types with $X = \{a, b\}$, $Y = \{c, d\}$, $Q = \{1, 2\}$ and $q_o = 1$; (i) deterministic Moore machine, (ii) deterministic Mealy machine, (iii) nondeterministic Moore machine, and (iv) nondeterministic Mealy machine.

Table 1) and is called *quantisation*. The deterministic actions at the microstate-level may appear stochastic at the macrostate-level since information is lost in the quantisation. One macrostate corresponds to an uncountable set of possible microstates (just in theory, in practice the RNN is simulated on a computer with finite precision) of microstates.

Another common ingredient of RNN-RE algorithms is *systematic testing* of the RNN with different inputs (from the domain or generated specifically for the extraction) and the (macro)states and outputs are stored and used to induce the finite state machine (ingredient two). The third ingredient (of Table 1) is the obvious machine construction, a process often concurrently executed with the state and output generation.

Many times, the generated machine is then minimised using a standard minimisation algorithm (Hopcroft & Ullman 1979), this is the fourth common ingredient of RNN-RE algorithms. FSM minimisation is however not part of all algorithms, and can also be considered as an external feature, independent of the actual extraction.

4 Evaluation criteria and taxonomy

To simplify comparisons and to structure the descriptions of the algorithms several evaluation criteria have been chosen. These criteria are used in the tables containing paper summaries, in Section 5, and are of central importance to the discussions in Section 7.

4.1 Main criteria

4.1.1 Rule type

As mentioned before (in Section 3.2) the rules generated by RNN-RE algorithms are FSMs that are either *deterministic*, *nondeterministic* or *stochastic*. They can also be in a Mealy or Moore format. In our classification of rule types we will also distinguish whether the machine (and underlying RNN) is producing a binary *accept/reject* decision at the end of a string (i.e. like an FSA) or if the task is produce an output sequence of symbols based on the input sequence (typically for prediction).

4.1.2 Quantisation

One of the most varying elements of existing RNN-RE algorithms is the *state space quantisation* method. Examples of methods used are: hierarchical clustering, vector quantisation and self organising maps (see Section 7.2 for a detailed discussion).

4.1.3 State generation

Another important criterion is the *state generation* procedure for which there are two basic methods: *searching* and *sampling*. These will be described further in the descriptions of the algorithms.

The rule type, quantisation method and state generation method can be considered to constitute the main distinguishing features of RNN-RE algorithms, and they have therefore been used to structure this survey.

4.1.4 Network type and domain

Although not a feature of the extraction algorithm *per se*, the *network type(s)* and in which *domain(s)* each RNN-RE algorithm has been used, will be explicitly listed for each presented technique.

4.2 Criteria from the ADT taxonomy

Andrews et al. (1995) introduced a taxonomy, the *ADT*[5] *taxonomy*, for RE algorithms which has since been an important framework when introducing new, or discussing existing, RE algorithms (e.g. Schellhammer, Diederich, Towsey & Brugman 1998, Vahed & Omlin 1999, Craven & Shavlik 1999, Blanco, Delgado & Pegalajar 2000). The five evaluation criteria in the ADT taxonomy were: expressive power, translucency, portability, rule quality and algorithmic complexity. The reason for not adopting the ADT taxonomy in this survey is that it was based on a broader scope than we have in this paper, as this paper focuses strictly on *recurrent* networks. For some of their classification aspects, all RNN-RE algorithms would end up in the same class and such classifications would therefore not be very informative. The ADT taxonomy does, however, provide us with some very useful viewpoints taken in Section 7.5. Some of the terminology from the ADT taxonomy will also appear in various places in this survey, therefore a brief description of the ADT aspects is given below.

4.2.1 Expressive power

The *expressive power* is basically the type of rules generated by the RE and hence subsumed by our rule type criteria. ADT identified (when taking also Tickle et al. (1997) and Tickle & Andrews (1998) into account) four basic classes:

- propositional logic (i.e. *if...then...else*)
- nonconventional logic (e.g. fuzzy logic)
- first-order logic (i.e. rules with quantifiers and variables), and
- finite state machines.

Almost all rules from RNN-RE algorithms would fall into the last category.

4.2.2 Translucency

One of the central aspects in the ADT taxonomy, *translucency*, described as the "degree to which the rule-extraction algorithm 'looks inside' the ANN" is less relevant here since it is not a distinguishing feature of RNN-RE algorithms. ADT initially identified three types of RE algorithms, (i) *decompositional* algorithms where rules are built on the level of individual neurons and then combined, (ii) *pedagogical* approaches using a black-box model of the underlying network and

[5] "ADT" comes from the names of the authors, Andrews, Diederich and Tickle.

(iii) *eclectic* algorithms with aspects from both previous types. Tickle & Andrews (1998) also introduced a fourth intermediate category, *compositional*, to accommodate for RNN-RE algorithms that are all (except for one pedagogical algorithm (Vahed & Omlin 1999, Vahed & Omlin 2004)) based on analysing ensembles of neurons (i.e. the hidden state space).

4.2.3 Portability

The *portability* describes how well an RE technique covers the set of available ANN architectures. As for translucency, the portability is probably much the same for all RNN-RE algorithms. It is also a quite complex aspect of RE techniques (tightly bound with translucency, and, in terms of feasibility, with algorithmic complexity) and we have therefore chosen not to distinguish RNN-RE algorithms by this criterion. The portability of the existing RNN-RE algorithms should however be quite high compared to RE algorithms in general since the requirements on the underlying RNN are not very restrictive. A deeper discussion of the portability of existing RNN-RE techniques is found in Section 7.5.3 where *implicit* demands of the underlying RNN/domain are identified.

4.2.4 Quality

The *quality* of the extracted rules is a very important aspect of RE techniques, and perhaps the most interesting for evaluation of the quality of the algorithms. This aspect differs from the other aspects in that it evaluates RE algorithms at the level of the *rules* rather than the level of the RE algorithms themselves.

Based on previous work, such as Towell & Shavlik (1993), four sub-aspects of rule quality were suggested in the ADT taxonomy;

- *rule accuracy*, i.e. the ability of the rules to generalise to unseen examples,
- *rule fidelity*, i.e. how well the rules mimic the behaviour of the RNN,
- *rule consistency*, i.e. the extent to which equivalent rules are extracted from different networks trained on the same task, and
- *rule comprehensibility*, i.e. readability of rules and/or the size of the rule set.

4.2.5 Algorithmic complexity

The *algorithmic complexity* of RE algorithms is unfortunately also often an open question as authors seldomly analyse this explicitly (cf. Andrews et al. (1995)). Although Golea (1996) showed that RE can be an NP-hard problem, it is unclear how existing heuristics affect the actual *expected* time and space requirements. For RNN-RE, the complexity issue has not received much attention and the issue is in itself quite complex as the execution time can be affected by many factors, e.g. number of state nodes, number of input symbols, granularity of the quantisation, RNN dynamics etc. Since the algorithmic complexity of most RNN-RE algorithms is an open question, we will not go into any details on this aspect in our taxonomy.

5 RNN-RE techniques

Although we have identified some common characteristics among the RE algorithms, dividing them into groups has been a painstaking task as there is an innumerable number of ways to do so. The techniques will be presented in a primarily chronological order and when a later technique is similar to an earlier, it will be presented in connection with its predecessor (although this relation may be constituted by coincidental similarities rather than a direct continuation of prior work).

First some early work that laid the ground for RE techniques to be developed will be presented in the next subsection. Then the algorithms will be described in more detail in subsections 5.2-5.7, but for fuller descriptions of the algorithms, we refer to the original papers.

5.1 Pre-RE approaches

To understand the roots of FSM extraction (which is the the primary form of RNN-RE) from recurrent networks, it is useful to recognise that in some early attempts to analyse RNNs, clustering techniques were used on the state space and clusters corresponding to the states of the FSM generating the language were found (clustering is still today one of the central issues of the research on RE from RNNs). Hierarchical Cluster Analysis (HCA) was used for analysing RNNs in a few early papers on RNNs (Cleeremans et al. 1989, Servan-Schreiber et al. 1989, Elman 1990, Servan-Schreiber et al. 1991). The authors found that for a network trained on strings generated by a small finite-state machine, the HCA may find clusters in the state space apparently corresponding to the states of the grammar. The clusters of the HCA were labelled using the labels of the states of the underlying state machine, making it easy to draw the connection between the RNN and the FSM.

The fact that much of the early research on RNNs was conducted on problem sets *explicitly based* on FSMs may have biased subsequent research to look for these FSMs inside the network. It is quite reasonable to assume that if you train a learning system to solve a problem with a known solution, you would be lead to search for structures of the known solution within the system.

However, for some successful networks (e.g. Servan-Schreiber et al. 1991), no clusters corresponding directly to the states of the FSM, which generated the training set language, were found. This meant that the network had an alternative, but apparently correct, representation of the problem, that differed from the one anticipated. This was probably due to the fact that the network did not necessarily need to mimic the *minimal* machine. That a non-minimal FSM was found was later shown to be the case when RE was used on RNNs (Giles, Miller, Chen, Chen & Sun 1992) where the resulting DFA was then minimised. Therefore FSM minimisation is included in most RNN-RE algorithms.

The basic problem of using only clustering (and not recording the transitions) for analysing RNNs is that there is no reliable way of telling how the clusters relate to each other temporally[6]. If the exact same FSM is not found, the

[6]There are other, more general problems of an HCA-based analysis of ANNs in general, as adjacent states (i.e. hidden unit activations) may be interpreted differently by the output layer and remote states may have the same interpretation (Sharkey & Jackson 1995). For RNNs this becomes even more problematic as the state is not only mapped into an output

clusters may not be labelled using the original FSM as a source and the temporal ordering of the clusters is therefore lost. This problem was also observed by Elman (1990): "the temporal relationship between states is lost. One would like to know what the trajectories between states [...] look like.". The solution of this problem led to the development of FSM extraction from RNNs.

5.2 Search in equipartitioned state space

The algorithm of Giles and collegues (Giles, Chen, Miller, Chen, Sun & Lee 1991, Giles, Miller, Chen, Chen & Sun 1992, Omlin & Giles 1996*b*) partitioned the state space into equally sized hypercubes (i.e. macrostates) and conducted a breadth-first search by feeding the network input patterns until no new partitions were visited. The transitions among the macrostates (induced by input patterns) were the basis for the extracted machine, see figure 4. The search started with a predefined initial state of the network and tested all possible input patterns on this microstate. The first encountered microstate of each macrostate was then used to induce new states. This guaranteed the extraction of a deterministic machine since any state *drift* (Das & Mozer 1994, Das & Mozer 1998) was avoided as the search was pruned when reentering already visited partitions. The extracted automaton was then minimised using a standard minimisation algorithm for DFAs (Hopcroft & Ullman 1979). The algorithm is summarised in Table 2.

DFA extraction, regular partitioning, breadth first search (Giles et al. 1991, Giles, Miller, Chen, Chen & Sun 1992, Omlin & Giles 1996*b*)	
Rule type:	Moore DFA with binary (accept/reject) output.
Quantisation:	Regular partitioning by q intervals in each state dimension, generating q^N bins of which typically only a small subset is visited by the RNN.
State generation:	Breadth-first search.
Network(s):	Predominantly used on second-order RNNs
Domain(s):	Predominantly regular languages with relatively few symbols. Some applied domains, e.g. quantised financial data (Giles, Lawrence & Tsoi 1997, Lawrence, Giles & Tsoi 1998, Giles, Lawrence & Tsoi 2001)

Table 2: Summary of algorithm extracting DFA through searching in an equipartitioned state space.

The central parameter of the algorithm is the quantisation parameter q of the equipartition. The authors suggested starting with $q = 2$ and increasing it until an automata consistent with the training set is extracted, i.e. the termination criteria is to have perfect accuracy of the rules. The choice of q is however usually not explicitly described as part of the RE algorithm (one exception is in the description by Omlin (2001) where the suggested incremental procedure is also part of the algorithm).

but also mapped recursively to all succeeding outputs through the state transitions.

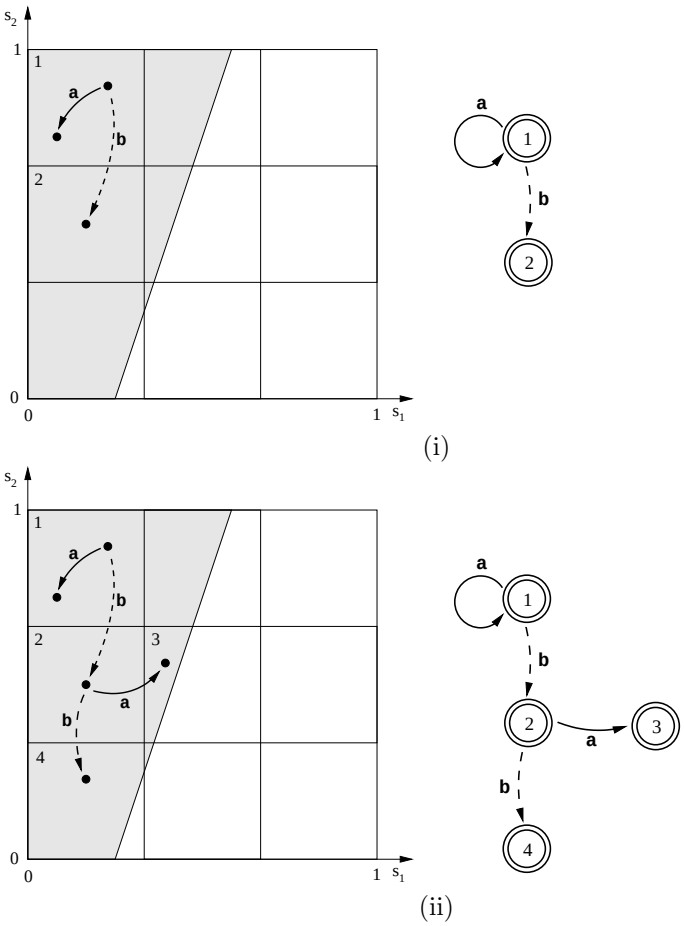

Figure 4: *Part one.*

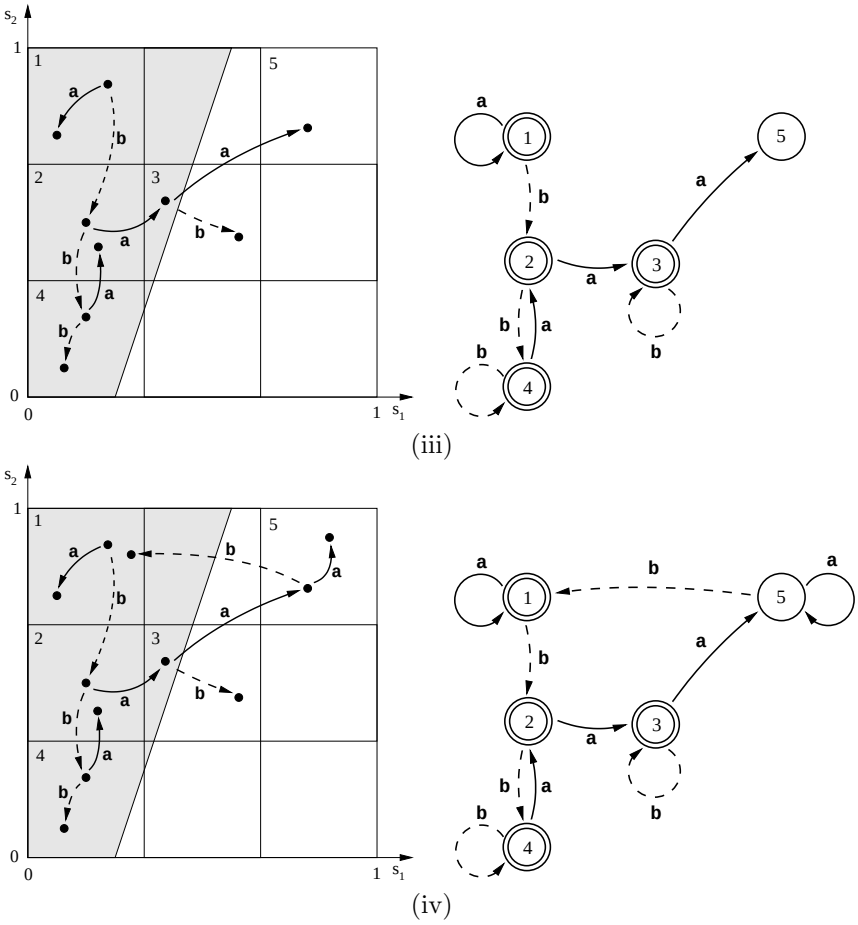

Figure 4: *Part two.* An example of the DFA extraction algorithm of Giles et al. (1991) used on an RNN with two state nodes trained on a binary language and the quantisation parameter $q = 3$. The state space is divided into an accept and reject region (gray and white respectively). The algorithm expands the graph until all nodes have two outgoing arcs. Note that the macrostate corresponding to node 3 could actually be interpreted both as an accept *and* reject state depending on the microstate, but the algorithm used the interpretation of the first encountered microstate as the interpretation of the macrostate, i.e. in this case accept.

Giles, Miller, Chen, Chen & Sun (1992) found that the generalisation ability of the extracted machines sometimes exceeded that of the underlying RNNs. Since the networks were trained on regular grammars, if the extraction result was a DFA equivalent with the original grammar that generated the training/test set, generalisation would also be perfect. Giles, Miller, Chen, Sun, Chen & Lee (1992) showed that during successful training of an RNN, the extracted DFA will eventually belong to the same *equivalence class* as the original DFA. Existence of equivalence classes over different degrees of quantisation (i.e. different values of q) was used in Omlin, Giles & Miller (1992) as an indicator of the networks' generalisation ability, i.e. if the extracted DFAs for increasing values of q collapsed into a single equivalence class, it was taken as a sign of good generalisation ability without the need for explicitly testing this on a separate test set.

The same algorithm has been used in various other contexts: as part of rule refinement techniques (e.g. Omlin & Giles 1992, Giles & Omlin 1993, Das et al. 1993, Omlin & Giles 1996*c*), as an indicator of underlying language class (Blair & Pollack 1997), as a method for complexity evaluation (e.g. Bakker & de Jong 2000), as part of a quantitative comparison of different RNN architectures (Miller & Giles 1993), as a means for FSM acquisition[7] (e.g. Giles, Horne & Lin 1995) or simply as an analysis tool of the RNN solutions[8] (e.g. Giles & Omlin 1994, Goudreau & Giles 1995, Giles et al. 1997, Lawrence et al. 1998, Lawrence, Giles & Fong 2000, Giles et al. 2001, Bakker 2004). The algorithm has also been used in the context of *recursive* networks (Maggini 1998).

An apparent problem with this technique is that the worst-case number of clusters grows exponentially with the number of state nodes N (q^N). The time needed for the breadth-first search will also grow exponentially with the number of possible input symbols. In practice, however, the number of visited states is much smaller than the number of possible states. But there is also a risk that, since all possible input patterns are tested for all visited states, transitions not occurring in the domain will be tested. In this way, states that the network would never visit in its interaction with the domain would be encountered, resulting in an unnecessarily complex FSM (Jacobsson & Ziemke 2003*b*).

This, the earliest of RNN-RE methods, is also the most widely spread algorithm. Almost all following papers where new RNN-RE techniques have been proposed cite Giles, Miller, Chen, Chen & Sun (1992). But often these papers do not contain citations to each other, giving the first impression of the field as less diverse than it actually is. Consequently there is a surprising variety of RE approaches, some of them seemingly developed independently of each other.

5.3 Search in state space partitioned through vector quantisation

An alternative to the simple equipartition quantisation was suggested already by Zeng et al. (1993) where a k-means algorithm was used to cluster the microstates. The centres of the clusters, the *model vectors*, were used as the basis for the breadth-first search, i.e. the RNN was tested with all input symbols for

[7] Implicitly, however, more or less *all* papers using RE are in some way on FSM/language acquisition. This division into RNN-RE usage should be taken with a grain of salt since each paper has more than one contribution.

[8] This is also implicitly part of many other papers as well.

each model vector state (cf. the equipartition algorithms where the first encountered RNN state is the basis for further search). See figure 5 for an illustrative example of this algorithm. A similar approach, also using k-means, developed seemingly independently from Zeng et al. (1993) is presented in Frasconi, Gori, Maggini & Soda (1996) and Gori, Maggini, Martinelli & Soda (1998), and a similar SOM-based approach in Blanco et al. (2000). A summary of these approaches are given in Table 3.

DFA extraction, vector quantifier, breadth first search (Zeng et al. 1993, Frasconi et al. 1996, Gori et al. 1998)	
Rule type:	Moore DFA with binary (accept/reject) output.
Quantisation:	k-means.
State generation:	Breadth-first search.
Network(s):	Second-order RNNs (Zeng et al. 1993), Recurrent radial basis function network, (Frasconi et al. 1996, Gori et al. 1998), RNN with an external pushdown automaton (Sun, Giles & Chen 1998).
Domain(s):	Regular binary languages (Tomita 1982), context free languages (Sun et al. 1998).

Table 3: Summary of algorithms extracting DFA through searching in a state space partitioned by vector quantisation.

The main difference between a vector-quantised (VQ) and equipartitioned state spaces, apart from partitions not being of equal sizes and shapes, is that the clusters are not fixed *prior* to the extraction but are instead *adapted* to fit the actually occurring state activations in the RNN. In principle, vector quantisation should be able to scale up to more state nodes than the equipartition method since the number of partitions can be arbitrarily selected independent from state space dimensionalty. But the downside is that the clustering may fail, or at least result in different clusters given different random initialisations of clusters prior to the adaptation of the clusters. The appropriate number of clusters, k, is also not easy to anticipate. Zeng et al. (1993) proposed a method to determine k automatically to some degree, but this method also needed user-defined parameters. The complexity problem associated with breadth-first search (described in the previous section) exists also for this vector-quantised approach, although the number of macrostates is more under control than for the equipartitioned approach.

To support an appropriate clustering of states, Zeng et al. (1993) and Frasconi et al. (1996) induced a bias for the RNN to form clusters during training. Other studies have also followed this approach (Das & Das 1991, Das & Mozer 1994, Das & Mozer 1998). In this way the RE algorithm and the RNN will become more entangled and if the altered training conditions of the underlying RNN are critical for successful RE, the algorithm would be less portable to other types of RNNs. RE techniques that can be used on already existing networks (i.e. typically not designed to be easy to analyse) are described by Tickle & Andrews (1998) as more attractive techniques.

In the presented search-based approaches, the reentering into partitions was

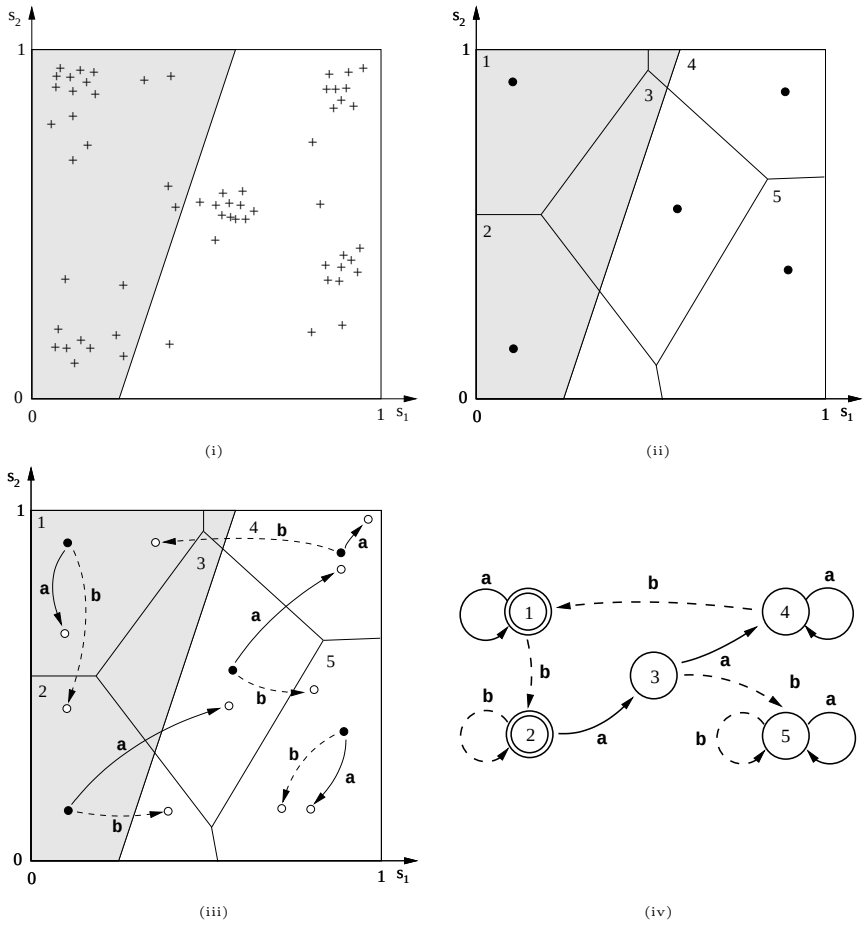

Figure 5: An illustrative example of rule extraction through breadth-first search in a state space clustered by k-means. (i) The states of the RNN are sampled during training, (ii) these states are clustered into a predefined number of clusters, (iii) a breadth-first search (cf. figure 4) is conducted based on the model vectors and (iv) the machine is constructed.

the basis of pruning the search. A different pruning strategy was suggested by Alquézar & Sanfeliu (1994*a*) and Sanfeliu & Alquézar (1995) who chose to use the domain to determine search depth (the algorithm is summarised in Table 4). A *prefix tree* (see figure 6) was built based on the occurrences of positive and negative strings in the training set, i.e. the prefix tree contained only strings present in the training set. The states of the RNN were generated using only the strings in the prefix tree. The authors used RE as part of their Active Grammatical Inference (AGI) learning methodology, an iterative rule refinement technique.

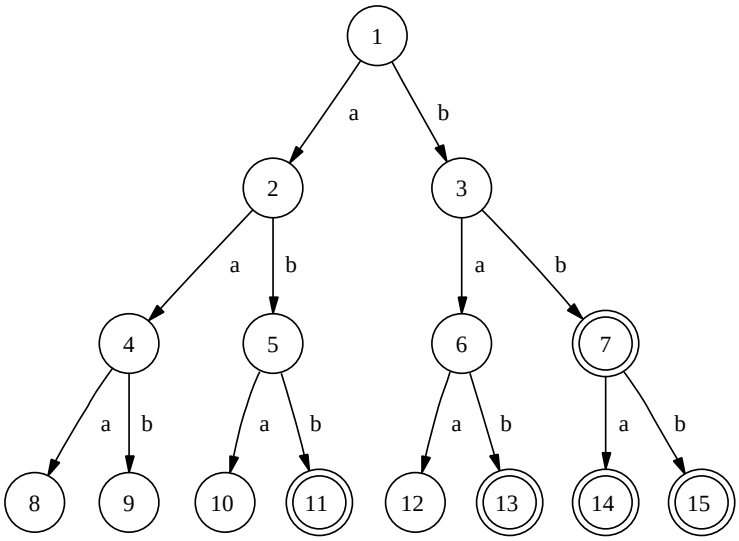

Figure 6: An example of a prefix tree of depth 3, created from a language that only accepts strings containing at least two `b`'s.

The states generated with the prefix tree were the basis of the initial machine. The spatially closest pair of these states was then merged iteratively until further clustering would result in an inconsistency. This RE technique was also used for a wide variety of regular grammars and two types of networks in (Alquézar et al. 1997). The authors reported that the extracted machines on average performed significantly better than the original RNNs.

5.4 Sampling-based extraction of DFA

Instead of conducting a search in the quantised state space, activity of the RNN in interaction with the data/environment can be recorded. In this way, the domain can be considered as heuristics confining the states of the RNN to only relevant states.

Already before RE techniques for RNNs were developed, sampling of the state space using the domain, was the most natural way to conduct analysis of RNNs (Cleeremans et al. 1989, Servan-Schreiber et al. 1989, Elman 1990). The first RE technique based on sampling the RNN was proposed by Watrous & Kuhn (1992) (see Table 5). The quantisation of the state space was based on splitting individual state units' activations into intervals. They described that

DFA extraction, hierchical clustering, sampling on domain (Alquézar & Sanfeliu 1994*a*), and (Sanfeliu & Alquézar 1995)	
Rule type:	Unbiased Moore DFA. Unbiased means the output is trinary (accept, reject and unknown).
Quantisation:	Hierarchical clustering.
State generation:	A prefix-tree is built based on the examples of the training set.
Network(s):	First-order RNN (not specified in Alquézar & Sanfeliu (1994*a*) but in Sanfeliu & Alquézar (1995)).
Domain(s):	At least 15 different regular binary languages (Alquézar et al. 1997).

Table 4: A summary of the search-based DFA extracting algorithm proposed by Alquezar and Sanfeliu for unbiased grammars.

these intervals could be merged and split to help the extraction of minimal and deterministic rules. The procedure of state splitting, however, is a bit unclear in their description and may require intervention from the user.

DFA extraction, dynamic interval clustering, sampling on domain (Watrous & Kuhn 1992)	
Rule type:	Moore DFA with binary (accept/reject) decision.
Quantisation:	Dynamically updated intervals for each state unit. States are collapsed and split through updating the intervals.
State generation:	Sampling the RNN while processing the domain.
Network(s):	Second-order RNNs.
Domain(s):	Regular binary languages (Tomita 1982).

Table 5: A summary of the sampling-based DFA extraction algorithm proposed by Watrous and Kuhn (1992).

Manolios & Fanelli (1994) chose to use a simple vector quantifier instead of dynamically updated intervals. Randomly initiated model vectors were repeatedly tested until a deterministic machine was found. The termination of this procedure is however not guaranteed. The algorithm is summarised in Table 6.

A similar approach was suggested in Tiňo & Šajda (1995) where an algorithm for removing inconsistent transitions was introduced. This algorithm could, however, fail under certain circumstances so that the extraction of a DFA could not be guaranteed. A star topology self-organising map (SOM, (Kohonen 1995)) was used to quantise the state space. Tiňo & Šajda (1995) were the first to extract Mealy instead of Moore machines and also the first who did not confine the output to binary accept/reject decisions (not counting the unbiased DFA of Alquézar & Sanfeliu (1994*a*)). This algorithm is summarised in Table 7.

The breadth-first search will reliably find consistent DFMs since the search is pruned before inconsistencies leading to indeterminism are introduced. The

DFA extraction, vector quantifier, sampling on domain (Manolios & Fanelli 1994), originally in tech. rep. (Fanelli 1993)	
Rule type:	Moore DFA with binary (accept/reject) decision.
Quantisation:	A simple vector quantifier, details unclear.
State generation:	Sampling on a test set.
Network(s):	First-order RNNs.
Domain(s):	Regular binary languages (Tomita 1982).

Table 6: The sampling-based DFA extractor proposed originally in Fanelli (1993).

DFM extraction, SOM, sampling on domain (Tiňo & Šajda 1995)	
Rule type:	Mealy DFM with multiple output symbols.
Quantisation:	Star topology SOM.
State generation:	Sampling on training set.
Network(s):	Second-order RNNs.
Domain(s):	Regular formal language domains with either two or three input symbols (not counting the end-of-string symbol) and two or three output symbols.

Table 7: Summary of the sampling-based DFM extractor of Tiňo and Šajda (1995)

DFM will also be complete since all symbols are tested on all states. In sampling the state space, determinism is no longer guaranteed since two microstates of the same macrostate may result in transitions, to different macrostates, triggered by the same symbol. Two state vectors in the same partition may also be mapped to different classes in the output. The extracted machines may also be *incomplete* since not all symbols may be tested on all states. Therefore the DFM extraction through sampling may fail as in the above cases of Watrous & Kuhn (1992), Manolios & Fanelli (1994) and Tiňo & Šajda (1995). It is unclear how incomplete machines were handled in the above described approaches and perhaps the extracted machines were small enough and domains simple enough to not result in any such problems.

An approach to solve the problem of indeterminism is to use transition frequencies to discard the least frequent of inconsistent transitions. This heuristic should in most cases solve the inconsistency without deviating much from the operation of the underlying RNN in the majority of the transitions. This simple procedure was proposed by Schellhammer et al. (1998) (summarised in Table 8). They also handled the problem of incomplete machines by creating transitions to a predefined "rescue state" to make the machine complete. These simplifications did not significantly reduce the performance of the DFM and the rescue state made it possible for the machines to make "guesses" about inputs that otherwise would not be possible to parse.

DFM extraction, vector quantizer, sampling on domain (Schellhammer et al. 1998)	
Rule type:	Mealy DFM with a "rescue state" used to make machine complete.
Quantisation:	k-means.
State generation:	Sampling on training set. Inconsistencies solved by discarding the least frequent of inconsistent transitions.
Network(s):	SRN.
Domain(s):	Natural language prediction task.

Table 8: Summary of the only sampling-based DFM extractor where inconsistencies and incompleteness are handled.

5.5 Stochastic machine extraction

As described in the previous section, the extraction of *deterministic* FSMs (DFMs) from RNNs through sampling is hampered by the fact that the quantisation of the state space may lead to inconsistencies in the macrostate transitions. These inconsistent transitions (and potentially state interpretations) will however follow some patterns and if all such transitions are counted they can be transcribed into a *stochastic machine*, i.e. a nondeterministic machine with probabilities associated with the transitions. The inconsistencies that ruin a DFM extraction may in other words contain informative probabilities that more accurately describe the RNN.

An algorithm for extraction of stochastic machines from RNNs was proposed by Tiňo & Vojtek (1998). The algorithm quantised the state space using a SOM (as also Tiňo & Šajda (1995) did). The generation of states and state transitions was divided into two phases; the "pre-test" phase, where the RNN was domain-driven, and a "self-driven" phase, where the output of the RNN was used as input in the next time-step (this RNN was trained to predict a long sequence of symbols). In Tiňo & Köteles (1999) (further described in Tiňo, Dorffner & Schittenkopf (2000)) the SOM was replaced with a dynamic cell structure (DCS, Bruske & Sommer (1995)), but otherwise the algorithm was the same (see the summary in Table 9).

The stochastic machines are possible to analyse in new interesting ways. The authors (Tiňo & Vojtek 1998, Tiňo & Köteles 1999), for example, used *entropy spectra* (Young & Crutchfield 1993) to compare the probabilities of strings generated by the RNNs with the probabilities of the strings in the original source. The results were interesting but there were no indications, in that paper, of how well the extracted machines corresponded to the network (i.e. rule fidelity) or how well they generalised on any test set[9] (i.e. rule accuracy). The comprehensibility of the extracted rules can also not be determined from these papers. Compared to deterministic machines, simplification and minimisation are likely to be significantly more complex for stochastic machines.

Stochastic machine extraction, SOM, sampling on domain (Tiňo & Vojtek 1998, Tiňo & Köteles 1999)	
Rule type:	Stochastic Mealy finite state machine.
Quantisation:	SOM (unspecified topology) in Tiňo & Vojtek (1998) and DCS in Tiňo & Köteles (1999)
State generation:	Two phases: Sampling on training set and "self-driven" RNN.
Network(s):	Primarily second-order RNNs.
Domain(s):	Prediction of (four) symbols generated from continuous chaotic laser data and a chaotic series of binary symbols generated with iterated logistic map function.

Table 9: Summary of approaches of RNN-RE for extraction of stochastic machines.

Another related approach is perhaps not rule extraction *per se* but can perhaps at least be termed a *partial* rule extraction algorithm. A "neural prediction machine" (NPM) is constructed in (Tiňo, Čerňanský & Beňušková 2004). The NPM predicts the next symbol given the state of the network, i.e. the state dynamics are handled by the RNN and not extracted at all (see a summary of this approach in Table 10).

5.6 A pedagogical approach

All previously described algorithms fall into the category *compositional* in ADT's translucency classification (see Section 4.2). There is to our knowledge only one

[9] Unless the entropy spectra analysis is considered a form of accuracy measurement.

Neural prediction machine, vector quantizer, sampling on domain (Tiňo et al. 2004)	
Rule type:	A "Neural Prediction Machine" (NPM) predicting the next output based on current state of the RNN. State transitions not modelled.
Quantisation:	k-means
State generation:	Sampling.
Network(s):	First-order RNN.
Domain(s):	Continuous chaotic laser data domain transformed to four symbols and recursive natural language domains.

Table 10: Neural Prediction Machines (NPMs) differ from the FSM ordinarily extracted from RNNs in that state transitions are not incorporated into the model.

algorithm that uses a *pedagogical* approach instead. Vahed and Omlin (Vahed & Omlin 1999, Vahed & Omlin 2004) used a machine learning method requiring only the input and the output to extract the machine, i.e. the internal state is ignored (see the summary in Table 11). The data used for extraction was based on all strings up to a given length. The input and output of the network was recorded and was fed to the polynomial-time "Trakhtenbrot-Barzdin" algorithm (Trakhtenbrot & Barzdin 1973).

It was also reported that this algorithm was more successful in returning correct DFAs than clustering-based algorithms (Giles, Miller, Chen, Sun, Chen & Lee 1992). Actually this seems to be the only paper that at all describes an experimental comparison of different RE techniques.

The machine learning algorithm they used is indeed of polynomial time complexity, given that a prefix tree (see figure 6) is available. But the size of the prefix tree up to a string length L is of complexity $O(n^L)$, where n is the number of symbols. As a consequence, this approach is likely to have some problems scaling up to more complex problems with more symbols.

DFA extraction, black-box model (Vahed & Omlin 1999, Vahed & Omlin 2004)	
Rule type:	Moore DFA with binary (accept/reject) output.
Quantisation:	N/A.
State generation:	All strings up to a certain length.
Network(s):	Second-order RNN.
Domain(s):	One randomly generated 10-state DFA.

Table 11: The only RNN-RE algorithm where the internal state of the RNN is not regarded during the extraction process.

5.7 RE-supporting RNN architectures

As mentioned before, clusters can be induced during training to support RE in later stages (Zeng et al. 1993, Frasconi et al. 1996). This was originally suggested in Das & Das (1991) and further developed in Das & Mozer (1994) and Das & Mozer (1998). Training to induce clusters results, if successfully performed, in RNNs that are *trivially* transformed to finite machines. Since the focus of this paper is on the details of the *extraction procedure*, more details of these approaches will not be included here.

6 RNN-RE, Fool's gold?

Kolen (1993) showed with some simple examples that some dynamic systems with real-valued state space (e.g. an RNN) cannot be described discretely without introducing peculiar results (cf. Kolen & Pollack (1995)). If you want to approximate the behaviour of a physical system with a real-valued state space as a discrete machine you will not only risk that the approximation might not be exact. A more profound effect of the approximation is that induced machines, from the same physical system, may belong to completely different classes of computational models, depending only on how the transformation from the real-valued space to a discrete approximation is conducted.

This critique strikes at the very heart of RNN-RE since the quantisation of the state space is a crucial element of these algorithms and RNN-RE was actually termed "Fool's gold" by the author (Kolen 1993). He pointed out that RNNs should be analysed as dynamical systems or more specifically iterated function systems (IFSs) rather than state machines.

There are some replies to this critique, though. One simple approach is to avoid the problem by not modelling transitions at all (Tiňo et al. 2004), or even not to quantise the state space at all (Vahed & Omlin 1999, Vahed & Omlin 2004). Another response to Kolen's critique is that extraction of a state machine from an RNN has been proven to work if the underlying RNN robustly model given a finite state machine (Casey 1996). However, this does not alleviate the fact that the language class (isomorphic to classes of computational models) for unknown RNNs cannot reliably be recognised. But at least there is a theoretical "guarantee" that if there is an FSM at "the bottom" of an RNN, it can always be extracted in principle.

The failure of rule extraction from an RNN could therefore be an indicator that the underlying RNN is not regular. One first step in this direction has been proposed by Blair & Pollack (1997). They used unbounded growth of the macrostate set under increased resolution of the equipartition quantisation method as an indicator of a nonregular underlying RNN.

If we limit ourselves to real world domains, RE will be operating on finite domains, making FSM interpretations theoretically possible at all times (although they may not be the minimal description of a domain). In fact, since the focus of RNN-RE research is on FSM extraction, the question should not be whether a language class is misjudged by an RE algorithm or not (since extraction on the level of the class of regular languages is one of the premises), but rather how well the extracted finite machine approximates the network, as proposed by Blair & Pollack (1997). How to evaluate the fidelity of an FSM and whether

this evaluation may distinguish between errors stemming from a poorly quantised state space or from a higher language class in the RNN/domain remains an open issue.

In summary, although Kolen's critique is justified, there are still reasons why further research on RE from RNNs is interesting. There is a lack of sophisticated analysis tools that can handle the complexity of RNNs, and RNN research is hampered by this fact. Although there are theoretical possibilities that ambiguous answers may be given about an RNN through RE, this holds also for many other analysis techniques. Kolen also did not provide any indications under *which conditions* RE would fail. There is a possibility that inconsistencies become less frequent for higher degrees of fidelity in the extraction process since Kolen described such inconsistencies only for the most coarse-grained of fidelities (two and three macrostates).

In other words, the theoretical disadvantages may turn out to be uncommon in real applications where any sophisticated analysis of the RNN is better than none. Of course, as for any analysis tool, the disadvantages must be kept in mind when examining the results.

7 Discussion

In this section, the described techniques will be summarised and evaluated from the perspectives of the evaluation criteria, rule type, quantisation method, state generation, network type and domain. The criteria in the well known ADT taxonomy (described in Section 4.2) will also be presented in more detail and RNN-RE discussed based on these criteria.

7.1 Rule types

It is quite clear that most of the research described in section 5 has been focused on extracting "traditional" DFA for classification of binary strings as grammatical/ungrammatical (Giles et al. 1991, Giles, Miller, Chen, Chen & Sun 1992, Watrous & Kuhn 1992, Zeng et al. 1993, Alquézar & Sanfeliu 1994*a*, Manolios & Fanelli 1994, Sanfeliu & Alquézar 1995, Omlin & Giles 1996*b*, Frasconi et al. 1996, Gori et al. 1998, Vahed & Omlin 1999, Vahed & Omlin 2004). Only a few DFA extraction algorithms are used on domains with more than two output symbols (Tiňo & Šajda 1995, Schellhammer et al. 1998). It is also interesting to notice that only three papers (Schellhammer et al. 1998, Tiňo & Vojtek 1998, Tiňo & Köteles 1999) have studied DFA RNN-RE in a prediction domain while prediction of sequences is a quite common approach in RNN research in general. (Elman 1990, Alquézar & Sanfeliu 1994*b*, Gers & Schmidhuber 2001, Jacobsson & Ziemke 2003*a*).

The crisp DFA do not model probabilistic properties of macrostate transitions and macrostate interpretations; that kind of information is lost in the rules, independently of whether search or sampling is used to generate states. Hence, a more expressive set of rules may be represented in stochastic FSM (Tiňo & Vojtek 1998, Tiňo & Köteles 1999) and the fidelity, i.e. the coherence of the rules with the RNN, of stochastic rules should in principle be higher (given the same premises, e.g. quantisation) than for their deterministic counterparts. The

fidelity can, however, be measured in various ways, as the term is not clearly defined, leading to possibly ambiguous results.

The choice between stochastic and deterministic rules is not obvious. Deterministic rules are easily analysed but may oversimplify and hide essential details of the underlying RNN and hide errors stemming from the chosen quantisation method. Given a certain state space quantisation, stochastic rules should on the other hand be a more accurate description of the underlying RNN than deterministic rules. This is because a deterministic RNN viewed through the "state space quantisation window" may appear indeterministic.

An approach to solve this may be to push further on the way chosen by Schellhammer et al. (1998) where probabilities were calculated and then used as heuristics for transforming the incomplete and nondeterministic machine into a deterministic and complete machine. That way the information loss from going from the RNN to a deterministic machine could possibly be tracked.

A last, "exotic", form of rules is the Neural Prediction Machine. The NPM is only predicting the output of the network given the state, but is not concerned with the internal mappings of states in the RNN (Tiňo et al. 2004).

7.2 State space quantisation

Clearly, there is no consensus about how to quantise the state space. Methods that have been used are (see Section 5 for more complete reference lists): regular (grid) partition (Giles et al. 1991), k-means (Zeng et al. 1993, Frasconi et al. 1996, Schellhammer et al. 1998, Tiňo et al. 2004, Cechin, Pechmann Simon & Stertz 2003), SOM (Tiňo & Šajda 1995, Tiňo & Vojtek 1998, Blanco et al. 2000), dynamical cell structures (Tiňo & Köteles 1999), "other" vector quantifiers (Manolios & Fanelli 1994), hierarchical clustering (Alquézar & Sanfeliu 1994*a*), dynamically updated intervals (Watrous & Kuhn 1992) and fuzzy clustering (Cechin et al. 2003). That makes eight different techniques, not counting small variations in implementations. Although just a fraction of existing clustering techniques have at all been tested (Mirkin 1996, Jain, Murty & Flynn 1999) it is clear that a multitude of existing clustering techniques has been used to solve the quantisation problem.

But what is most striking about this multitude of various techniques used is not that they are so many, but that there are *no studies* comparing different quantisation techniques to each other in the context of RNN-RE. How should they be evaluated then? What characterises a good quantisation in the context of RNN-RE? It is not necessarily spatial requirements (Sharkey & Jackson 1995), as is usually the case for evaluation of clustering techniques (Jain et al. 1999), but rather requirements based on properties of the extracted rule set. To have clusters that are *spatially* coherent and well separated is of less importance than the *fidelity* of the resulting rules. If an evaluation method for quantisation techniques could be defined in the context of RNN-RE it could help the research on RNN-RE to find better techniques with clustering techniques tailor-made for the purposes of extracting good rules (where the definition of "good" of course depends on the goal of the RE).

7.3 State generation

There are two basic strategies for generating the states in the RNN (see Section 5 for more complete reference lists): *searching* (Giles et al. 1991, Zeng et al. 1993, Frasconi et al. 1996) and *sampling* (Watrous & Kuhn 1992, Manolios & Fanelli 1994, Alquézar & Sanfeliu 1994*a*, Tiňo & Šajda 1995, Schellhammer et al. 1998, Tiňo & Vojtek 1998, Tiňo et al. 2004).

As for the clustering techniques, there are no studies comparing searching- and sampling-based RNN-RE experimentally, apart from one preliminary study (Jacobsson & Ziemke 2003*b*). Unlike for clustering techniques, however, it is quite easy to see at least a few of the consequences of the choice of state generation method.

First of all, breadth-first search will obviously have problems of scaling up to larger problems. The search-based techniques should be especially sensitive to the number of input symbols. There are also reasons to believe that for prediction networks in domains that are not completely random, many of the transitions and states generated with breadth-first search would not be relevant or ever occur in the domain (Jacobsson & Ziemke 2003*b*). Machines extracted with search are however guaranteed to be deterministic which may very well be desired (see discussion in the previous section). The extraction is also guaranteed to end up with a *complete* machine where all possible inputs are tested on all encountered states.

RE through sampling on the domain is not guaranteed to result in deterministic machines. If a deterministic machine is required, there is no guarantee that a certain state space quantisation will result in a solution since inconsistencies might occur. The other problem, as mentioned before, is the possibility that not all states will have recordings of all possible outgoing transitions, i.e. the machine will be incomplete. Suggestions to solutions to both these problem have only been proposed in one paper (Schellhammer et al. 1998). Sampling-based RE techniques may therefore be a better strategy for extraction of stochastic rather than deterministic machines.

RE through sampling is, however, RE not only from the RNN, but also from the domain. No rules other than those that apply to the domain will be extracted as may be the case for breadth-first search, e.g. if two inputs never occur in succession in a certain prediction task, there is no point in extracting rules from the network about such events. The domain sample provides heuristics about which rules are interesting to explore. The machine will then possibly give a good insight into the domain as well as the RNN, since the presence and absence of transitions in the machines are a result of the domain. If an incomplete prediction machine is extracted it could in principle also be used to detect anomalies in a test sequence, since unexpected inputs (i.e. inputs at states where the outgoing transition is not in the model) could be used to trigger warnings. If, however, the domain sample, used for rule induction, is lacking some crucial element, the extracted rules will also be incomplete with respect to this element.

7.4 Network types and domains

The networks that have been studied using RNN-RE are in most cases relatively small networks with few hidden nodes. This may be due to the fact that most

domains used were simple enough to allow small networks to be trained.

There are also significantly more second-order than first-order networks. Probably, this is an effect of the focus on formal language domains where second-order RNNs are more commonly used than first-order networks (Goudreau, Giles, Chakradhar & Chen 1994).

The proportion of RNN architectures that have not been tested using RE is very high, perhaps due to scalability problems in current RE techniques, or that analysis of the internal behaviour is less an issue than the actual performance of the RNNs.

As mentioned in Section 7.1, the investigated domains are mostly requiring only binary string classification. More complex domains with many symbols or deep syntactical structures, etc., have as yet not been tested using RNN-RE. Therefore, the applicability of these techniques is to a large degree an open issue. The importance of rule extraction as described by Andrews et al. (1995), e.g. explanation capability, verification of ANN components, etc., is therefore lesser than it would have been if the techniques had been demonstrated to work on the state-of-the-art RNNs operating on the most challenging domains.

7.5 Relation to the ADT taxonomy

Although the ADT taxonomy (Andrews et al. 1995, Tickle et al. 1997, Tickle & Andrews 1998) has not been used explicitly to classify the techniques in the taxonomy of this paper, it can be highly useful as a basis for discussion of the techniques. Some important points will be evident when viewing RNN-RE in terms of these criteria (see Section 4.2 for descriptions of the criteria).

7.5.1 Expressive power

In the ADT taxonomy, basically all rules from RNN-RE algorithms fall under the same category ("finite state machine"). Therefore, more RNN-RE characteristic features were chosen to discriminate between the rule types (cf. Section 7.1).

7.5.2 Translucency

Although translucency may have been the most central aspect in the ADT taxonomy it has been of less importance to discriminate RNN-RE techniques since (almost) all techniques fall under the same category (compositional). Therefore translucency may not be very informative for *comparing* RNN-RE techniques, but it can be interesting to study what RNN-RE algorithms actually require of the underlying network. This is, however, also highly related to the portability of RNN-RE techniques which will be discussed next.

7.5.3 Portability

Even though most RNN-RE algorithms are compositional and have the, in principle, same requirements on the underlying RNN there are some implicit requirements that could be interesting to bring up, especially if the existing RNN-RE algorithms are to be applied on other RNN architectures and other domains than so far. Current RE techniques are preferably used on RNNs that:

1. operate in discrete time, i.e. continuous-time RNN will not work. There is however no known work on continuous time RNNs in the domain of FSM generated languages (Forcada & Carrasco 2001),
2. have a clearly defined input, state and output, i.e. less or randomly structured RNNs may be problematic (e.g. echo state networks (Jaeger 2003)),
3. have a fully observable state, otherwise unobserved state nodes or noise in the observation process would disturb the extraction process since the state space would not be reliably quantised,
4. have state nodes that can be set explicitly (for search-based techniques),
5. are deterministic, otherwise the same problem as if the state is not fully observable would occur,
6. have certain dynamical characteristics, e.g. sampling-based extraction of a deterministic machines operates preferably on non-fractal structures of the state space to be feasible,
7. are fixed during RE, i.e. no training can be allowed during the RE process.

Due to algorithmic complexity, the underlying RNN must also not have too many state nodes (especially when using regular partitioning, which does not scale up well). The domain should also not contain too many possible input patterns (symbols), especially if breadth-first search is used. The domain should preferably also be discrete (or be transformed to a discrete representation prior to RE (e.g. Giles et al. 1997)) since there are no means for representing continuous input data in the current types of extracted rules. Therefore the problem of good quantisation of input (and perhaps output) space may also be interesting to study in relation to RE. A continuous but quantised input stream may also give the illusion of an underlying indeterministic RNN, since one *input symbol* then may be the result of several *input patterns*, just as several microstates may lead to the observation of the same macrostate.

Of course, a list such as the one above needs to be taken with a grain of salt. The problem of making a list of implicit requirements is just that they are *implicit* (i.e. not readily apparent). There may therefore be other essential requirements that we have not managed to figure out at this stage. Also, the strengths of these requirements are not clear either, some of them may actually be quite easily alleviated with some enhancements of current RNN-RE techniques.

7.5.4 Quality

As mentioned before (cf. Section 4.2.4), the quality of RNN-RE techniques is (or should be) evaluated at the level of the actual rules, rather than at the level of the algorithms. Extracted rules depend not only on the algorithm but also on the underlying domain and network. Evaluation of the rule quality therefore requires extensive studies comparing different RE techniques under similar conditions. Unfortunately, such studies have not yet been conducted for most RNN-RE algorithms.

A discussion of rule quality is further complicated by two things: firstly, as already mentioned, rule quality can only be reliably evaluated on the extracted

rules themselves; and secondly, but perhaps more importantly, the *goal* of the extraction needs to be clearly specified before the quality can be evaluated. In some papers it is clear that the accuracy is the most important aspect of rule quality. Accuracy is a good means for evaluating rule quality as long as the goal for rule extraction is to find rules that are "as good as possible" on the domain (see Figure 7 for a visual illustration of the relation between accuracy and fidelity). But if the goal is to examine the network's ability to generalise (e.g. for software verification), accuracy should not be used. After all, if the network is tested with an accuracy-maximising RE method, the result may be rules with a performance better than the network (a result confirmed by many studies). Therefore, for purposes of RNN analysis, *fidelity* should be the preferred quality evaluation criteria.

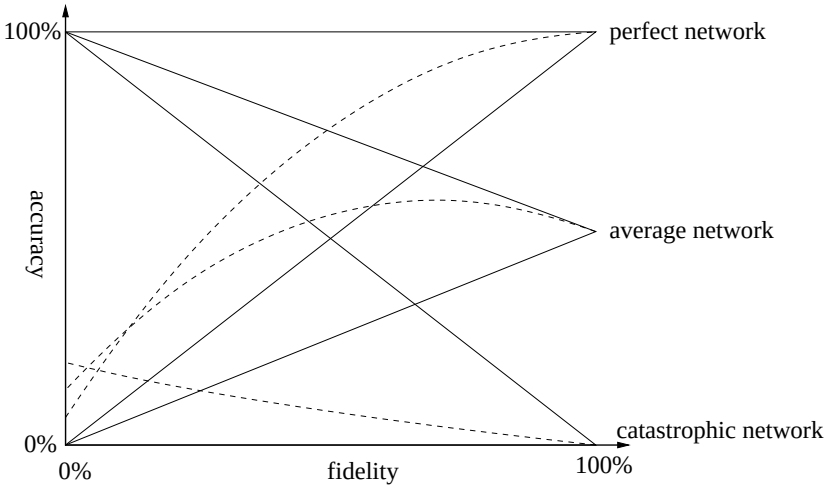

Figure 7: An illustration of the relation between the two quality criteria accuracy and fidelity for networks with different performances. The higher the fidelity, the more the accuracy of the rules matches the performance of the network. For lower fidelity, the accuracy of the rules may divert more from the performance of the network (indicated by the solid lines). Typically, however, lower fidelity would mean lower accuracy (typical example indicated by dashed lines), since there are more erroneous rules than correct ones. This would however not be true if the extraction mechanism included measures to, apart from extracting the rules from the RNN, tweak them to fit the data more accurately.

There are, in the existing corpus of papers on RNN-RE, some indirect results that provide some indications for some of the rule quality sub-categories, accuracy, fidelity, consistency and comprehensibility.

Some studies indicate that the extracted machines indeed have high *accuracy* since they may even be generalising better than the underlying RNN (Giles, Miller, Chen, Chen & Sun 1992, Giles, Miller, Chen, Chen & Sun 1992, Giles & Omlin 1993, Omlin & Giles 1996*b*). There are, however, unfortunately no studies where the *fidelity* of the extracted rules has been tested separately from the accuracy. The studies tend to focus on networks that are quite successful on their domain and under such circumstances the difference between fidelity and

accuracy is very small. For networks performing badly on their domain, high fidelity would, however, imply low accuracy since the errors of the network then would be replicated by the machine (cf. Figure 7).

Rule consistency has not been studied extensively although some papers touch the subject. Rules extracted from a network during training were found to fall under a sequence of equivalence classes during training (Giles, Miller, Chen, Sun, Chen & Lee 1992). This can be seen as an example of consistency since the extracted rules after a certain time of training eventually stabilised in the same equivalence class, i.e. the set of quite similar networks at the later part of the training resulted in equivalent rules. The consistency over different parameter settings (of the quantisation parameter q in the equipartitioned RNN-RE algorithm) has also been proposed as an indicator of regularity in the underlying network (Blair & Pollack 1997). These results on consistency are, however, more or less indirect.

Rule comprehensibility has not been experimentally compared in any way, but it is clear that it is an important issue to ensure further progress for RNN-RE research. After all, if the goal of extracting rules is to understand the underlying incomprehensible network, the rules should preferably be comprehensible themselves. The extracted rules have been informative in some qualitative way in the reviewed papers but in some cases it is clear that a higher degree of fidelity/accuracy reduces the possibility to find easily understandable results (Lawrence et al. 2000). Standard DFA minimisation techniques have already been used to reduce the DFAs to their minimal representation. But further heuristics (such as suggested by Schellhammer et al. (1998)) for how rules can be made more comprehensible are needed. Algorithms with a user-defined preference in the "comprehensibility/fidelity tradeoff" (Craven & Shavlik 1999) have also been pointed out as a direction for future research.

7.5.5 Algorithmic complexity

The algorithmic complexity of RNN-RE algorithms is one of the most unexplored and least documented aspects. To seek further answers about the algorithmic complexity of RNN-RE algorithms could, however, be very fruitful. A deeper understanding of the role of dynamics in the underlying RNN would for example be needed to determine the complexity of individual RNNs. Important answers and even more important new questions would definitely be the result of breaking down these algorithms until the complexity becomes clear.

The algorithmic complexity of the extraction may however be inherently linked to the complexity of the domain and the intrinsic computational complexity of the RNN as an instance of a physical computing device (Crutchfield 1993). The latter form of complexity is also inherently dependent on the observation process itself (Kolen & Pollack 1995). In other words, the complexity issue is inherently complicated and may turn out to be too difficult to evaluate by other means than trial and error. It is however clear that, since RNN-RE is inherently a computationally demanding process, heuristics for more efficient RNN-RE algorithms will remain one of the central issues in future RNN-RE development.

8 Open issues

Throughout this paper, open issues have been hinted at, especially in Section 7. In this section these open issues are summarised and extrapolated. First some general questions will be posed concerning how our current state of knowledge about existing techniques may be enhanced, then future directions will be outlined for how the usability and strengths of RNN-RE algorithms can be improved.

8.1 Development of scientific tools and measures

First of all, it is quite clear that some basic questions have not yet been asked. One such basic question is: What are the possible goals of RNN-RE? Possible answers could be (taken partly from Andrews et al. (1995) in the context of RE in general):

1. to acquire a generic model of the domain, i.e. the RNN is used merely as a tool in the acquisition process (data mining),
2. to provide an explanation of the RNN,
3. to allow verification/validation of the RNN with respect to some requirements (cf. software testing) and thus make new, potentially safety critical, domains possible for RNNs,
4. to improve on current RNN architectures by pinpointing errors,
5. to integrate the rules (or at least state space quantisation) with the RNN (Zeng et al. 1993, Frasconi et al. 1996).

The appropriate measure to evaluate the success of a specific instance of an RNN-RE algorithm being applied on an RNN (and domain) depends highly on which of these (or other) goals are desired. For the first goal, for example, the maximisation of accuracy is the prime target. For the other goals the maximisation of fidelity is likely to be more important. In some cases, however, comprehensibility may be crucial. In other cases, it is imaginable that the efficiency of the algorithm (in terms of execution time or required memory storage) is the primary objective. There may also be other, more domain specific measures to evaluate the degree of goal achievement.

Some of these terms; fidelity, accuracy, comprehensibility, are in turn not completely defined more than at an intuitive level. If two techniques are to be quantitatively compared in terms of any of these criteria, they need to be more clearly defined (perhaps such definition may be domain dependent and there may also be room for more than one definition of, e.g., fidelity).

When it comes to fidelity, for example, one can divide the term into *domain specific fidelity* and *domain independent fidelity*, the former of which is only concerned with the rule-RNN agreement under the interaction with the domain and the latter defined more generally, over all possible input interactions. The definition of consistency may also be refined further by differentiating between consistency over different RNNs, sample sets (used during RE), different initial states in the same RNN, or RNN-RE parameter settings etc.

After these terms of rule quality have been more clearly defined, there is still a need for a methodical cross-comparison of RNN-RE algorithms. For example, benchmark RNN models could be developed for this purpose. This would make comparisons more feasible since different researchers can make sure that the underlying system (the RNN-domain combination) is held constant and only the studied aspect, e.g. clustering methods, is varied. Such benchmarks could be made available in a publicly available database[10] which would help not only research on RNN-RE algorithms but also RNN/ANN research in general[11].

8.2 Deeper investigations of existing algorithms

The existing algorithms have not been tested against each other, except that the only pedagogical approach (Vahed & Omlin 1999, Vahed & Omlin 2004) has been experimentally compared with the equipartitioned, search-based approach (Giles, Miller, Chen, Sun, Chen & Lee 1992). Although one may argue that in order to make progress, new techniques must be developed rather than sticking to the old solutions, we argue that in order to take the right decisions about how to find better algorithms, we need to better understand the old solutions before creating new and exotic ones.

For example, at this point we do not have any information about what quantisation method seems most promising, or whether stochastic or deterministic machines should be preferred. This may be largely due to the lack of a clear goal or well defined measures to compare the algorithms with. Another central issue that has not been tested is how these algorithms scale up. The scalability issue is addressed in Craven & Shavlik (1999) where a broader definition, not only referring to algorithmic complexity, was suggested: scalability in terms of algorithmic complexity *and* comprehensibility. Whatever the definition, however, the scalability of current techniques has not been evaluated to any significant extent.

The list of implicit requirements discussed in Section 7.5.3 is a starting point in mapping out what current techniques can and cannot do. But the list is very limited and not at all supported by any deeper experimental or theoretical verification. A suggestion is to map the "limits" of current techniques by applying them to architectures and domains on the borderline of what the techniques should be able to master.

8.3 Utilisation of RNN-RE algorithms

Something that would drive the development of any analysis tool are new applications. The algorithms described in this survey have been primarily used on quite small domains and networks. By using existing RNN-RE algorithms on more demanding domains, two effects could be expected; 1. it would open up research on domains that are more "interesting" in themselves, and 2. weaknesses of existing algorithms would become apparent, which would generate interesting directions for how to improve them. For example, some algorithms are focused

[10] The idea behind this database was born in discussions with André Grüning as a solution to the difficulty and effort of reproducing RNNs. An analysis of the possible design and required infrastructure of such a database has been initiated by the author.

[11] A similar database approach is proposed by Kolen & Pollack (1991), who suggested a database where the initial weights, prior to training, would be saved.

primarily on string classification tasks and are perhaps not portable to other tasks (such as symbol prediction tasks), since the assumption of the existence of a fully defined binary classification may not hold in other tasks.

We have also noted that most RNN types have not yet been considered in RNN-RE research, e.g. Long Short-Term Memory (Hochreiter & Schmidhuber 1997) and Echo State Networks (Jaeger 2003). These networks may not be more complex for RE *per se* but still have properties putting different sorts of requirements on the RNN-RE algorithms.

Apart from just a straight progression towards more complex domains and networks, other possible uses for RNN-RE can be prospected:

- A striking thing in the previous employment of RNN-RE algorithms is that almost all papers dealt exclusively with perfect or almost perfect networks, i.e. networks very well adapted to their domains. In some cases, perfect generalisation capability of the extracted rules was part of the termination criteria (Giles, Miller, Chen, Chen & Sun 1992). For RNN-RE to be an interesting tool for RNN researchers, however, the successful analysis of "failing" RNNs is important to be able to improve upon these failures. Such RE-based failure analysis may be far more powerful than just pointing out the existence of errors and may be integrated as part of the training of the network.

- Incomplete machines extracted through domain interaction could be used to trigger warnings for new "exotic" input patterns that have not been encountered during the extraction (cf. discussion in end of Section 7.3).

- A distance metric between RNNs could be defined by comparing rules extracted by RNN-RE. RNNs are otherwise difficult to compare directly since completely different weights can give equivalent behaviour and small differences in weights may result also in very different behaviours. This sort of distance metric would be favourable when using RNN ensembles (Sharkey 1996) to ensure a heterogeneous set of RNNs in the ensemble.

- The idea that RNN-RE can be used as an indication of the complexity of the underlying RNN and domain could be further developed. Previous studies seem to show promising results (Crutchfield & Young 1990, Blair & Pollack 1997, Bakker & de Jong 2000) with regards to complexity estimations that go beyond Shannon entropy and Kolmogorov complexity.

8.4 RNN-RE enhancement and development

There are certainly an innumerable number of ways the existing techniques can be enhanced. The goals for enhancement can be found among some of the criteria of the ADT-taxonomy: portability, rule quality and algorithmic complexity. Portability can be enhanced in the sense that in the algorithms after some additional progress can be applied to more domains and network types. The rule quality can be improved (after first defining exactly what is meant by rule quality) by focusing development on rule accuracy, fidelity, consistency and/or comprehensibility. The algorithmic complexity and scalability can be addressed by developing more refined heuristics.

8.4.1 Future goals/demands of RNN-RE algorithms

What should we expect from future algorithms? Some ideas are given by Craven & Shavlik (1999) where methods for controlling the "comprehensibility/fidelity tradeoff" are identified as an important line of research. This "tradeoff" issue may be expanded to techniques where the user may, through the setting of a few parameters, not only have the ability to choose between fidelity and comprehensibility, but also fidelity and accuracy, fidelity and computation time etc. (Figure 8). The multiple choice of goals could result in a wide variety of RNN-RE algorithms, each adapted for their specific purpose. But, as suggested by Craven & Shavlik (1999), a single algorithm where the user may choose between, e.g. fidelity and comprehensibility should be the most preferable.

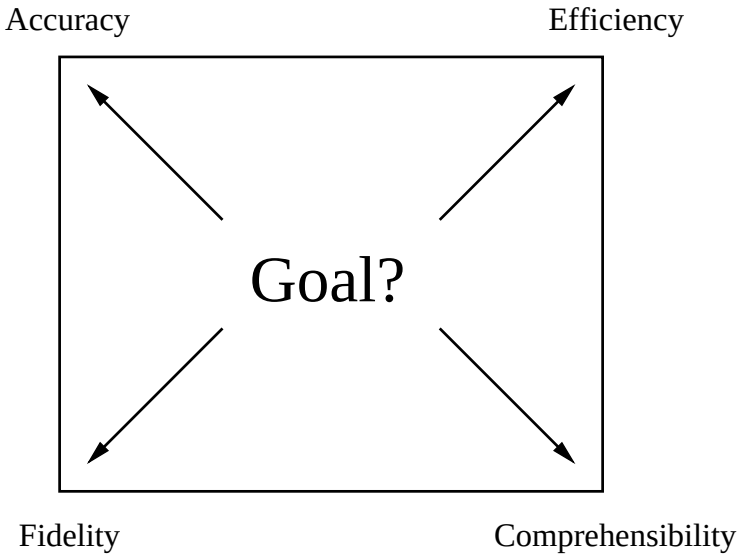

Figure 8: Four, possibly opposite, goals of RNN-RE that in an ideal algorithm would be simply be chosen by the setting of a few user-defined parameters.

In an ideal RNN-RE algorithm the relation between execution time, fidelity and comprehensibility may be as illustrated in Figure 9. Rules should be refined gradually over time and the more time available, the higher the possibility of acquiring rules of high fidelity and/or comprehensibility ("anytime rule extraction" (Craven & Shavlik 1999)).

To ensure comprehensibility, rules could be presented in more than one format. Therefore transformation to other rule formats, e.g. regular expressions, can be interesting to study. These transformations need not to be loss-free, perhaps thereby enhancing readability by creating smaller rule sets. The rule transformations may also be goal-oriented to express certain sub-rules that may be of special interest to the user.

In general, RNN-RE algorithms should have the possibility of being used in a goal-oriented manner, e.g. to let the RE focus on rules for an isolated event or output symbol or sequence of symbols. For example, find the sequence of inputs that will cause prediction of a critical event in an RNN. The algorithm

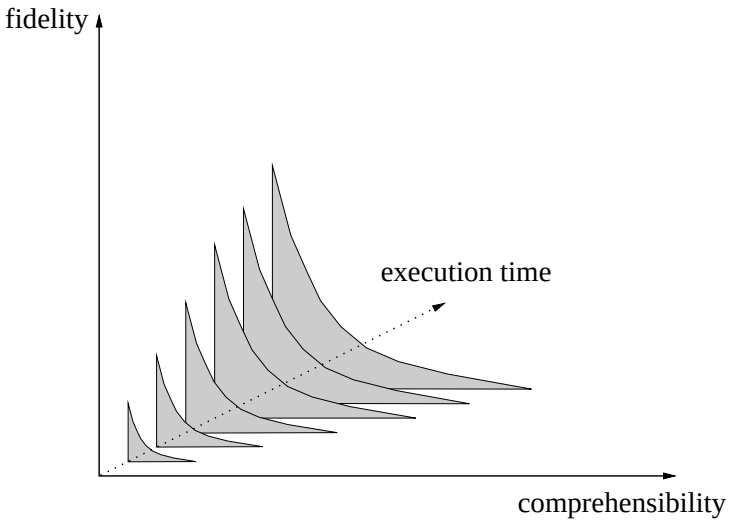

Figure 9: The relation between execution time, fidelity and comprehensibility for ideal RNN-RE algorithms with possible gradual refinement of the rules. The more time available, the more the degree of freedom in choosing between high fidelity and comprehensibility.

should then strive for maximisation of rule quality with respect to this specific goal, while omitting others.

8.4.2 Other possible enhancements

The goals described above are important general goals. But other more detailed enhancements of RNN-RE algorithms can also be suggested:

- Clustering techniques where the quality of the rules is the primary goal, rather than the grouping of spatially related data points, should be developed. The current clustering techniques used are not specifically designed for RNN-RE.
- The above can perhaps be achieved through the following line of reasoning: At the microstate level, the underlying network is completely deterministic and the quantisation should be defined such that this property is satisfied also at the macrostate level (for sampling-based extraction of stochastic machines). One way to achieve this is to have determinism-maximisation through gradual refinement of the quantisation function. Gradual refinement in this case means merging and splitting macrostates according to their context in the extracted machine[12].
- Such merging and splitting of macrostates may be achieved through the integration of quantisation and rule minimisation algorithm which may

[12]This idea was also born in discussions with André Grüning and preliminary results show that information-theoretic tools, such as conditional entropy, can be appropriate to use for this purpose.

give efficient means of refining the quantisation since macrostates can be merged through the equivalence of their corresponding states in the final machine and split if a state has nondeterministic outgoing transitions.

- Another method for achieving deterministic rules was proposed by Schellhammer et al. (1998) where stochastic rules were simply transformed into deterministic using simple heuristics. This method could be further developed to minimise the fidelity loss in the transformation. Thereby we would get the best out of the sampling (i.e. the efficiency) and the search techniques (i.e. deterministic rules).

- To extract discrete rules from RNNs operating in continuous domains the RNN-RE-technique should be integrated with input- and output-space quantisation (preferably with readable rules for these spaces, e.g. in if-then-else format).

- Rule extraction using slightly randomly distorted datasets as heuristics could help solving the problem of extracting rules (when using sampling) that only applies strictly to the domain sample that was used during extraction.

- A more controlled version of the above is to do "re-extraction" of uncertain/infrequent but possibly important rules by querying the network (Craven & Shavlik 1994). This can, for example, be done by directly setting states in the network to be in the vicinity of the model vector (or something equivalent) of the macrostate of interest and then testing the effect of feeding the RNN various possible inputs.

- The underlying task of the RNN (e.g. prediction) can be integrated into the rules in order to identify more exactly when and how erroneous behaviour occur in the network. This can be done simply by marking which states in the extracted machines are involved in the errors. This error can perhaps then be further traced back, in the rules, to the actual erroneous behaviour.

- The above can be further integrated with the training of the network by letting the trainer update the weights only in situations (microstates) identified by the rules as being part of an erroneous behaviour (or macrostate).

- The RNN distance metric (suggested in Section 8.3) could be implemented by first aligning the states of two machines extracted from two RNNs according to their equivalence. The unaligned states can then be used to evaluate the difference between the models (using for example relative entropy of possible future output patterns to measure the difference between states).

8.5 Some more suggestions

Since this paper is, in part, aimed at attracting more researchers to the field it is perhaps a good idea to not just identify open issues, but also to give some practical hints about what should be done. These hints are actually basically a repetition of Craven & Shavlik (1999), but they are important enough to be repeated.

First of all, if you are developing a new RNN-RE algorithm, make sure to strive for generality (i.e. high portability). The usefulness of the algorithm you develop will directly correlate with how easily it can be used on RNNs already developed, implemented and tested, originally without intentions of making them suitable for RE. In fact, Craven & Shavlik (1999) even suggested that the RE algorithms should be made so general that not even the assumption that the underlying system is a neural network at all is necessary. In fact, some things indicate that this is already a fact for most RNN-RE algorithms, considering the very limited assumptions of the underlying RNN (cf. Section 3.1).

Another good piece of advice is to seek out collaborators who already have RNNs they want to analyse (Craven & Shavlik 1999). It is highly unlikely that you will have time to develop both state-of-the-art RNNs and state-of-the-art RNN-RE algorithms at the same time. Finding willing collaborators should not be too difficult since researchers applying novel RNNs on new domains will most likely benefit from the knowledge acquired through rule extraction.

Another important ingredient for making a technique attractive is to make implementations of the techniques publicly available (Craven & Shavlik 1999). After all, the techniques are aimed at being used by researchers that are quite busy in developing their own line of work.

9 Conclusions

Ideally, if the research RNN-RE community had been really successful, these techniques would be among the first analysis techniques to be used when new RNN architectures had been developed or a new domain had been concurred. But we are not there yet.

Nowadays, despite numerous achievements, there seems to be no apparent common direction in the RNN-RE research community. In most cases, developed algorithms do not seem to be built based on previous results and there seems to be a very slow (if any) progress towards handling more complex RNNs and domains that RNN-RE algorithms can handle. In fact, only one algorithm has been used in any wide extent in the follow-up work, and moreover, it is the *first* RNN-RE algorithm developed (Giles, Miller, Chen, Chen & Sun 1992). It is surprising that it has not been replaced by anything significantly better in the years that have passed since then. Actually, later algorithms may very well be better, but they are still not used as frequently as the first one, and there are almost no comparative studies.

So, in summary, the RNN-RE research (and on other RNN analysis tools) seems to be an active, but slowly progressing, field. It seems as if more focus is still on developing more efficient training algorithms and more complex or efficient RNN architectures than on analysis of the end-results of these more sophisticated approaches.

In this paper we hope to have identified important research issues that need to be addressed to help give this field a push forward. As discussed in Section 2, RNNs, and neural networks in general, are, as they are simulated entities, very "studyable" once we have tools to study them. And the algorithms reviewed in this paper may hold the seed for a deeper and more general notion of analysis than seen before. Better analysis tools may in turn help RNN research to progress more rapidly once we get a deeper understanding of what the networks

are actually doing.

After all, in many other disciplines of science, the quantum leaps in progress often stem from more sophisticated analysis tools and measuring devices producing qualitatively new data conflicting with existing models (anomalies) that eventually may result in scientific revolutions (Kuhn 1962). Today we have deep, but partially conflicting theories, of what the RNNs will be able to do in practice (i.e. the Turing machine equivalence vs. the difficulty to acquire correct behaviour through learning), but we have no means for evaluating what instances of RNNs are actually doing in an efficient manner. This situation can, for example, be compared with the situation prior to the discovery of the background radiation; some theories were predicting that the universe may be expanding from a very dense state (the Big Bang theory), but it was not until the microwave antenna could measure this radiation that this theory could be verified and selected among many others as a good description of the actual events.

Similar situations are found in the development of the telescope, the microscope, the particle accelerator (that actually discovered particles no one had predicted) and space flight etc. Tools without which we would have a much more limited understanding of our physical world.

With critical eyes, rule extraction from recurrent neural networks may seem an infinitesimal subfield within another infinitesimal subfield and thereby with a very limited potential to deliver interesting scientific results. But if there is a future microscope for zooming in on RNNs, we would hold that there are good reasons to believe that rule extraction mechanisms will be the operational parts, or "lenses", of that microscope. And as any real-world microscope, this RNN-microscope will, if general enough, be able to zoom in on other types of dynamical systems and physical computing devices and thus contribute to the scientific community in a considerably broader sense.

Acknowledgements

First of all I would like to thank my librarian Karin Lundberg for her efficient and tireless efforts of providing me with all papers, without her, this survey would not have been possible. I would also like to thank André Grüning, Amanda Sharkey, Ron Sun and Tom Ziemke for commenting on early versions of this paper and also Gunnar Buason, Anders Jacobsson and Claudina Riguetti for proofreading a later version. I must also thank several of the cited authors for helping me make the list of references more complete and for engaging in some very interesting email discussions.

References

Alquézar, R. & Sanfeliu, A. (1994*a*), A hybrid connectionist symbolic approach to regular grammar inference based on neural learning and hierarchical clustering, in '*Proceedings of ICGI'94*', pp. 203–211.

Alquézar, R. & Sanfeliu, A. (1994*b*), Inference and recognition of regular grammars by training recurrent neural networks to learn the next-symbol pre-

diction task, in '*Advances in Pattern Recognition and Applications*', World Scientific Pub., pp. 48–59.

Alquézar, R., Sanfeliu, A. & Sainz, M. (1997), Experimental assessment of connectionist regular inference from positive and negative examples, in '*VII Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*', Vol. 1, pp. 49–54.

Andrews, R., Diederich, J. & Tickle, A. (1995), 'Survey and critique of techniques for extracting rules from trained artificial neural networks', *Knowledge Based Systems* **8**(6), 373–389.

Bakker, B. (2004), *The State of Mind: Reinforcement Learning with Recurrent Neural Networks*, Phd thesis, Unit of Cognitive Psychology, Leiden University.

Bakker, B. & de Jong, M. (2000), The epsilon state count, in J. A. Meyer, A. Berthoz, D. Floreano, H. Roitblat & S. Wilson, eds, '*From Animals to Animats 6: Proceedings of The Sixth International Conference on Simulation of Adaptive Behavior*', MIT Press, Cambridge, MA, pp. 51–60.

Barreto, G. A., Araújo, A. F. R. & Kremer, S. C. (2003), 'A taxonomy for spatiotemporal connectionist networks revisited: The unsupervised case', *Neural Computation* **15**, 1255–1320.

Bengio, Y., Simard, P. & Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE Transactions on Neural Networks* **5**(2), 157–166.

Blair, A. & Pollack, J. (1997), 'Analysis of dynamical recognizers', *Neural Computation* **9**(5), 1127–1142.

Blanco, A., Delgado, M. & Pegalajar, M. C. (2000), 'Extracting rules from a (fuzzy/crisp) recurrent neural network using a self-organizing map', *International Journal of Intelligent Systems* **15**, 595–621.

Blank, D. & 24 co-authors (1999), 'Connectionist symbol processing: Dead or alive?', *Neural Computing Surveys* **2**, 1–40.

Boden, M., Jacobsson, H. & Ziemke, T. (2000), Evolving context-free language predictors, in '*Proceedings of the Genetic and Evolutionary Computation Conference*', Morgan Kaufmann, pp. 1033–1040.

Bodén, M., Wiles, J., Tonkes, B. & Blair, A. (1999), Learning to predict a context-free language: Analysis of dynamics in recurrent hidden units, in '*Proceedings of ICANN 99*', IEEE, Edinburgh, pp. 359–364.

Bruske, J. & Sommer, G. (1995), 'Dynamic cell structure learns perfectly topology preserving map', *Neural Computation* **7**, 845–865.

Bullinaria, J. A. (1997), Analyzing the internal representations of trained artificial neural networks, in A. Browne, ed., '*Neural Network Analysis, Architectures and Applications*', IOP Publishing, pp. 3–26.

Carrasco, R. C. & Forcada, M. L. (2001), 'Simple strategies to encode tree automata in sigmoid recursive neural networks', *IEEE Transactions on Knowledge and Data Engineering* **13**(2), 148–156.

Carrasco, R. C., Forcada, M. L., Muñoz, M. A. V. & Ñeco, R. P. (2000), 'Stable encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units', *Neural Computation* **12**(9), 2129–2174.

Casey, M. (1996), 'The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction', *Neural Computation* **8**(6), 1135–1178.

Cechin, A. L., Pechmann Simon, D. R. & Stertz, K. (2003), State automata extraction from recurrent neural nets using k-means and fuzzy clustering, in '*XXIII International Conference of the Chilean Computer Science Society*', IEEE Computer Society, pp. 73–78.

Cicchello, O. & Kremer, S. C. (2003), 'Inducing grammars from sparse data sets: A survey of algorithms and results', *Journal of Machine Learning Research* **4**.

Cleeremans, A., McClelland, J. L. & Servan-Schreiber, D. (1989), 'Finite state automata and simple recurrent networks', *Neural Computation* **1**, 372–381.

Craven, M. C. & Shavlik, J. W. (1994), Using sampling and queries to extract rules from trained neural networks, in W. W. Cohen & H. Hirsh, eds, '*Machine Learning: Proceedings of the Eleventh International Conference*', Morgan Kaufmann, San Fransisco, CA.

Craven, M. W. & Shavlik, J. W. (1996), 'Extracting tree-structured representations of trained networks', *Advances in Neural Information Processing Systems* **8**, 24–30.

Craven, M. W. & Shavlik, J. W. (1999), Rule extraction: Where do we go from here?, Technical Report Machine Learning Research Group Working Paper 99-1, Department of Computer Sciences, University of Wisconsin.

Crutchfield, J. P. (1993), Observing complexity and the complexity of observation, in H. Atmanspacher, ed., '*Inside versus Outside*', Series in Synergetics, Springer, Berlin, pp. 235–272.

Crutchfield, J. P. (1994), 'The calculi of emergence: Computation, dynamics, and induction', *Physica D* **75**, 11–54.

Crutchfield, J. & Young, K. (1990), Computation at the onset of chaos, in W. Zurek, ed., '*Complexity, Entropy and the Physics of Information*', Addison-Wesley, Reading, MA.

Das, S. & Das, R. (1991), 'Induction of discrete-state machine by stabilizing a simple recurrent network using clustering', *Computer Science and Informatics* **21**(2), 35–40.

Das, S., Giles, C. L. & Sun, G. Z. (1993), Using prior knowledge in a NNPDA to learn context-free languages, in S. J. Hanson, J. D. Cowan & C. L. Giles, eds, '*Advances in Neural Information Processing Systems*', Vol. 5, Morgan Kaufmann, San Mateo, CA, pp. 65–72.

Das, S. & Mozer, M. (1998), 'Dynamic on-line clustering and state extraction: an approach to symbolic learning', *Neural Networks* **11**(1), 53–64.

Das, S. & Mozer, M. C. (1994), A unified gradient-descent/clustering architecture for finite state machine induction, in J. D. Cowan, G. Tesauro & J. Alspector, eds, '*Advances in Neural Information Processing Systems*', Vol. 6, Morgan Kaufmann Publishers, Inc., pp. 19–26.

Elman, J. L. (1990), 'Finding structure in time', *Cognitive Science* **14**, 179–211.

Fanelli, R. (1993), Grammatical inference and approximation of finite automata by elman type recurrent neural networks trained with full forward error propagation, Technical Report NNRG930628A, Dept. of Physics, Brooklyn College of the City University of New York.

Forcada, M. L. (2002), Neural networks: Automata and formal models of computation. An unfinished survey.
URL: *http://www.dlsi.ua.es/˜ mlf/nnafmc/*

Forcada, M. L. & Carrasco, R. C. (2001), Finite-state computation in analog neural networks: steps towards biologically plausible models?, in S. Wermter, J. Austin & D. Willshaw, eds, '*Emergent Computational Models Based on Neuroscience*', Lecture Notes in Computer Science, Springer-Verlag, pp. 482–486.

Frasconi, P., Gori, M., Maggini, M. & Soda, G. (1996), 'Representation of finite state automata in recurrent radial basis function networks', *Machine Learning* **23**(1), 5–32.

Gers, F. A. & Schmidhuber, J. (2001), 'Lstm recurrent networks learn simple context free and context sensitive languages', *IEEE Transactions on Neural Networks* .

Giles, C. L., Chen, D., Miller, C., Chen, H., Sun, G. & Lee, Y. (1991), Second-order recurrent neural networks for grammatical inference, in '*Proceedings of International Joint Conference on Neural Networks*', Vol. 2, IEEE Publication, Seattle, Washington, pp. 273–281.

Giles, C. L., Horne, B. G. & Lin, T. (1995), 'Learning a class of large finite state machines with a recurrent neural network', *Neural Networks* **8**(9), 1359–1365.

Giles, C. L., Lawrence, S. & Tsoi, A. (1997), Rule inference for financial prediction using recurrent neural networks, in '*Proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering (CIFEr)*', IEEE, Piscataway, NJ, pp. 253–259.

Giles, C. L., Lawrence, S. & Tsoi, A. C. (2001), 'Noisy time series prediction using a recurrent neural network and grammatical inference', *Machine Learning* **44**(1/2), 161–183.

Giles, C. L., Miller, C. B., Chen, D., Chen, H. H. & Sun, G. Z. (1992), 'Learning and extracting finite state automata with second-order recurrent neural networks', *Neural Computation* **4**(3), 393–405.

Giles, C. L., Miller, C. B., Chen, D., Sun, G. Z., Chen, H. H. & Lee, Y. C. (1992), Extracting and learning an unknown grammar with recurrent neural networks, in J. E. Moody, S. J. Hanson & R. P. Lippmann, eds, '*Advances in Neural Information Processing Systems*', Vol. 4, Morgan Kaufmann Publishers, Inc., pp. 317–324.

Giles, C. L. & Omlin, C. W. (1993), 'Extraction, insertion and refinement of symbolic rules in dynamically driven recurrent neural networks', *Connection Science* **5**(3 – 4), 307–337.

Giles, C. L. & Omlin, C. W. (1994), 'Pruning recurrent neural networks for improved generalization performance', *IEEE Transactions on Neural Networks* **5**(5), 848–851.

Golea, M. (1996), On the complexity of rule extraction from neural networks and network-querying, Technical report, Australian National University, Canberra, Australia.

Gori, M., Maggini, M., Martinelli, E. & Soda, G. (1998), 'Inductive inference from noisy examples using the hybrid finite state filter', *IEEE Transactions on Neural Networks* **9**(3), 571–575.

Gori, M., Maggini, M. & Soda, G. (1994), Scheduling of modular architectures for inductive inference of regular grammars, in '*ECAI'94 Workshop on Combining Symbolic and Connectionist Processing, Amsterdam*', Wiley, pp. 78–87.

Goudreau, M. W. & Giles, C. L. (1995), 'Using recurrent neural networks to learn the structure of interconnection networks', *Neural Networks* **8**(5), 793–804.

Goudreau, M. W., Giles, C. L., Chakradhar, S. T. & Chen, D. (1994), 'First-order vs. second-order single layer recurrent neural networks', *IEEE Trans. on Neural Networks* **5**(3), 511–518.

Hammer, B. & Tiňo, P. (2003), 'Recurrent neural networks with small weights implement definite memory machines', *Neural Computation* **15**(8), 1897–1929.

Hinton, G. E. (1990), 'Mapping part-whole hierarchies into connectionist networks', *Artificial Intelligence* **46**(1–2), 47–75.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural Computation* **9**(8), 1735–1780.

Hopcroft, J. & Ullman, J. D. (1979), *Introduction to Automata Theory, Languages, and Compilation*, Addison-Wesley Publishing Company.

Horne, B. G. & Giles, C. L. (1995), An experimental comparison of recurrent neural networks, in G. Tesauro, D. Touretzky & T. Leen, eds, '*Advances in Neural Information Processing Systems 7*', MIT Press, pp. 697–704.

Horne, B. G. & Hush, D. R. (1994), Bounds on the complexity of recurrent neural network implementations of finite state machines, in J. D. Cowan, G. Tesauro & J. Alspector, eds, '*Advances in Neural Information Processing Systems*', Vol. 6, Morgan Kaufmann Publishers, Inc., pp. 359–366.

Husbands, P., Harvey, I. & Cliff, D. T. (1995), 'Circle in the round: State space attractors for evolved sighted robots', *Robotics and Autonomous Systems* **15**(1-2), 83–106.

Jacobsson, H. & Ziemke, T. (2003*a*), 'Improving procedures for evaluation of connectionist context-free language predictors', *IEEE Transactions on Neural Networks* **14**(4), 963–966.

Jacobsson, H. & Ziemke, T. (2003*b*), Reducing complexity of rule extraction from prediction RNNs through domain interaction, Technical Report HS-IDA-TR-03-007, Department of Computer Science, University of Skövde, Sweden.

Jaeger, H. (2003), Adaptive nonlinear system identification with echo state networks, in S. T. S. Becker & K. Obermayer, eds, '*Advances in Neural Information Processing Systems 15*', MIT Press, Cambridge, MA, pp. 593–600.

Jain, A. K., Murty, M. N. & Flynn, P. J. (1999), 'Data clustering: A review', *ACM Computing Surveys* **31**(3), 264–323.

Kohonen, T. (1995), *Self-Organizing Maps*, Springer, Berlin, Heidelberg.

Kolen, J. F. (1993), Fool's gold: Extracting finite state machines from recurrent network dynamics, in J. Cowan, G. Tesauro & J. Alspector, eds, '*Neural Information Processing Systems 6*', Morgan Kaufmann, San Francisco, CA, pp. 501–508.

Kolen, J. F. (1994), *Exploring the Computational Capabilities of recurrent neural networks*, PhD thesis, The Ohio State University, Department of Computer and Information Sciences.

Kolen, J. F. & Kremer, S. C., eds (2001), *A Field Guide to Dynamical Recurrent Networks*, IEEE Press.

Kolen, J. F. & Pollack, J. B. (1991), Back propagation is sensitive to initial conditions, in R. P. Lippmann, J. E. Moody & D. S. Touretzky, eds, '*Advances in Neural Information Processing Systems*', Vol. 3, Morgan Kaufmann Publishers, Inc., pp. 860–867.

Kolen, J. & Pollack, J. (1995), 'The observers' paradox: Apparent computational complexity in physical systems', *Journal of Exp. and Theoret. Artificial Intelligence* **7**(3).

Kremer, S. C. (2001), 'Spatiotemporal connectionist networks: A taxonomy and review', *Neural Computation* **13**(2), 248–306.

Kuhn, T. S. (1962), *The structure of scientific revolutions*, University of Chicago Press, Chicago.

Lawrence, S., Giles, C. L. & Fong, S. (2000), 'Natural language grammatical inference with recurrent neural networks', *IEEE Transactions on Knowledge and Data Engineering* **12**(1), 126–140.

Lawrence, S., Giles, C. L. & Tsoi, A. C. (1998), Symbolic conversion, grammatical inference and rule extraction for foreign exchange rate prediction, in A. P. N. R. Y. Abu-Mostafa, A. S. Weigend, ed., '*Neural Networks in the Capital Markets NNCM96*', World Scientific Press, Singapore, pp. 333–345.

Linåker, F. & Jacobsson, H. (2001), Mobile robot learning of delayed response tasks through event extraction: A solution to the road sign problem and beyond, in B. Nebel, ed., '*Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI-2001*', Morgan Kaufmann, pp. 777–782.

Maggini, M. (1998), Recursive neural networks and automata, in C. L. Giles & M. Gori, eds, '*Adaptive Processing of Sequences and Data Structures*', Springer-Verlag, pp. 248–295.

Manolios, P. & Fanelli, R. (1994), 'First order recurrent neural networks and deterministic finite state automata', *Neural Computation* **6**(6), 1155–1173.

McCulloch, W. S. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics* **5**, 115–133.

Medler, D. (1998), 'A brief history of connectionism', *Neural Computing Surveys* **1**(1), 61–101.

Meeden, L. A. (1996), 'An incremental approach to developing intelligent neural network controllers for robots', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **26**(3), 474–85.

Miller, C. B. & Giles, C. L. (1993), 'Experimental comparison of the effect of order in recurrent neural networks', *International Journal of Pattern Recognition and Artificial Intelligence* **7**(4), 849–872.

Minsky, M. (1967), *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, NJ, chapter Neural Networks. Automata Made up of Parts.

Mirkin, B. (1996), *Mathematical Classification and Clustering*, Vol. 11 of *Nonconvex Optimization and Its Applications*, Kluwer.

Niklasson, L. & Bodén, M. (1997), Representing structure and structured representations in connectionist networks, in A. Browne, ed., '*Neural Network Perspectives on Cognition and Adaptive Robotics*', IOP Press, pp. 20–50.

Omlin, C. W. (2001), Understanding and explaining DRN behaviour, in J. F. Kolen & S. C. Kremer, eds, '*A Field Guide to Dynamical Recurrent Networks*', IEEE Press, pp. 207–228.

Omlin, C. W. & Giles, C. L. (1992), Training second-order recurrent neural networks using hints, in D. Sleeman & P. Edwards, eds, '*Proceedings of the Ninth International Conference on Machine Learning*', Morgan Kaufmann Publishers, San Mateo, CA, pp. 363–368.

Omlin, C. W. & Giles, C. L. (1996*a*), 'Constructing deterministic finite-state automata in recurrent neural networks', *Journal of the ACM* **43**, 937–972.

Omlin, C. W. & Giles, C. L. (1996*b*), 'Extraction of rules from discrete-time recurrent neural networks', *Neural Networks* **9**(1), 41–51.

Omlin, C. W. & Giles, C. L. (1996*c*), 'Rule revision with recurrent neural networks', *Knowledge and Data Engineering* **8**(1), 183–188.

Omlin, C. W. & Giles, C. L. (2000), Symbolic knowledge representation in recurrent neural networks: Insights from theoretical models of computation, in I. Cloete & J. M. Zurada, eds, '*Knowledge-Based Neurocomputing*', MIT Press.

Omlin, C. W., Giles, C. & Miller, C. (1992), Heuristics for the extraction of rules from discrete-time recurrent neural networks, in '*Proceedings of the International Joint Conference on Neural Networks*', Vol. I, pp. 33–38.

Omlin, C. W., Thornber, K. K. & Giles, C. L. (1998), 'Deterministic fuzzy finite state automata can be deterministically encoded into recurrent neural networks', *IEEE Transactions on Fuzzy Systems* **6**(1), 76–89.

Pollack, J. B. (1987), Cascaded back-propagation on dynamic connectionist networks, in '*Proceedings of the 9th Annual Conference of the Cognitive Science Society*', Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 391–404.

Rabin, M. O. (1963), 'Probabilistic automata', *Information and Control* **6**, 230–245.

Rodriguez, P. F. (1999), *Mathematical Foundations of Simple Recurrent Neural Networks in Language Processing*, PhD thesis, Univerity of California, San Diego.

Rodriguez, P., Wiles, J. & Elman, J. L. (1999), 'A recurrent network that learns to count', *Connection Science* **11**, 5–40.

Sanfeliu, A. & Alquézar, R. (1995), Active grammatical inference: a new learning methodology, in '*Shape, Structure and Pattern Recognition*', 5th IAPR International Workshop on Structural and Syntactic Pattern Recognition, World Scientific Pub., pp. 191–200.

Schellhammer, I., Diederich, J., Towsey, M. & Brugman, C. (1998), Knowledge extraction and recurrent neural networks: An analysis of an Elman network trained on a natural language learning task, in D. M. W. Powers, ed., '*Proceedings of the Joint Conference on New Methods in Language Processing and Computational Natural Language Learning: NeMLaP3/CoNLL98*', Association for Computational Linguistics, Somerset, New Jersey, pp. 73–78.

Servan-Schreiber, D., Cleeremans, A. & McClelland, J. L. (1989), Learning sequential structure in simple recurrent networks, in D. S. Touretzky, ed., '*Advances in Neural Information Processing Systems*', Vol. 1, Morgan Kaufmann, San Mateo, CA, pp. 643–652.

Servan-Schreiber, D., Cleeremans, A. & McClelland, J. L. (1991), 'Graded state machines: The representation of temporal contingencies in simple recurrent networks', *Machine Learning* **7**, 161–193.

Sharkey, A. J. C., ed. (1996), *Special Issue - Combining Artificial Neural Nets: Ensemble Approaches*, Vol. **8**(3/4) of *Connection Science*.

Sharkey, N. E. & Jackson, S. A. (1995), An internal report for connectionists, in R. Sun & L. A. Bookman, eds, '*Computational Architectures integrating Neural and Symbolic Processes*', Kluwer, Boston, pp. 223–244.

Siegelmann, H. T. & Sontag, E. D. (1995), 'On the computational power of neural nets', *Journal of Computer and System Sciences* **50**(1), 132–150.

Sima, J. & Orponen, P. (2003), 'General purpose computation with neural networks: a survey of complexity theoretic results', *Neural Computation* **15**, 2727–2778.

Sun, G. Z., Giles, C. L. & Chen, H. H. (1998), The neural network pushdown automation: Architecture, dynamics and learning, in C. Giles & M. Gori, eds, '*Adaptive Processing of Sequences and Data Structures*', number 1387 *in* 'Lecture Notes in Artificial Intelligence', Springer, pp. 296–345.

Sun, R. & Giles, C. L., eds (2001), *Sequence Learning: Paradigms, Algorithms, and Applications*, Vol. 1828 of *Lecture Notes in Artificial Intelligence*, Springer.

Sun, R., Peterson, T. & Sessions, C. (2001), The extraction of planning knowledge from reinforcement learning neural networks, in '*Proceedings of WIRN'2001*', Springer-Verlag, Heidelberg, Germany.

Tabor, W. & Tanenhaus, M. (1999), 'Dynamical models of sentence processing', *Cognitive Science* **24**(4), 491–515.

Tickle, A., Andrews, R., Golea, M. & Diederich, J. (1997), Rule extraction from artificial neural networks, in A. Browne, ed., '*Neural Network Analysis, Architectures and Applications*', IOP Publishing, pp. 61–99.

Tickle, A. B. & Andrews, R. (1998), 'The truth will come to light: directions and challenges in extracting the knowledge embedded within mined artificial neural networks', *IEEE Transactions on Neural Networks* **9**(6), 1057–1068.

Tiňo, P., Dorffner, G. & Schittenkopf, C. (2000), Understanding state space organization in recurrent neural networks with iterative function systems dynamics, in S. Wermter & R. Sun, eds, '*Hybrid Neural Symbolic Integration*', Springer Verlag, pp. 256–270.

Tiňo, P. & Hammer, B. (2003), 'Architectural bias in recurrent neural networks - fractal analysis', *Neural Computation* **15**(8), 1931–1957.

Tiňo, P., Horne, B. G., Giles, C. L. & Collingwood, P. C. (1998), Finite state machines and recurrent neural networks – automata and dynamical systems approaches, in J. E. Dayhoff & O. Omidvar, eds, '*Neural Networks and Pattern Recognition*', Academic Press, pp. 171–220.

Tiňo, P. & Köteles, M. (1999), 'Extracting finite-state representations from recurrent neural networks trained on chaotic symbolic sequences', *IEEE Trans. Neural Networks* **10**(2), 284–302.

Tiňo, P., Čerňanský, M. & Beňušková, L. (2004), 'Markovian architectural bias of recurrent neural networks', *IEEE Transactions on Neural Networks* pp. 6–15.

Tiňo, P. & Vojtek, V. (1998), 'Extracting stochastic machines from recurrent neural networks trained on complex symbolic sequences', *Neural Network World* **8**(5), 517–530.

Tiňo, P. & Šajda, J. (1995), 'Learning and extracting initial mealy automata with a modular neural network model', *Neural Computation* **7**(4), 822–844.

Tomita, M. (1982), Dynamic construction of finite-state automata from examples using hillclimbing, in '*Proceedings of Fourth Annual Cognitive Science Conference*', Ann Arbor, MI, pp. 105–108.

Tonkes, B., Blair, A. & Wiles, J. (1998), Inductive bias in context-free language learning, in '*Proceedings of the Ninth Australian Conference on Neural Networks*'.

Tonkes, B. & Wiles, J. (1999), Learning a context-free task with a recurrent neural network: An analysis of stability, in R. Heath, B. Hayes, A. Heathcote & C. Hooker, eds, '*Dynamical Cognitive Science: Proceedings of the Fourth Biennial Conference of the Australasian Cognitive Science Society*'.

Towell, G. G. & Shavlik, J. W. (1993), 'The extraction of refined rules from knowledge-based neural networks', *Machine Learning* **13**(1), 17–101.

Trakhtenbrot, B. A. & Barzdin, J. M. (1973), *Finite automata: behavior and synthesis*, Fundamental studies in computer science, North-Holland, Amsterdam.

Vahed, A. & Omlin, C. W. (1999), Rule extraction from recurrent neural networks using a symbolic machine learning algorithm, in '*6th International Conference on Neural Information Processing*'.

Vahed, A. & Omlin, C. W. (2004), 'A machine learning method for extracting symbolic knowledge from recurrent neural networks', *Neural Computation* **16**, 59–71.

von Neumann, J. (1956), Probabilistic logics and the synthesis of reliable organisms from unreliable components, in '*Automata Studies*', Princeton University Press, Princeton, pp. 43–98.

Watrous, R. L. & Kuhn, G. M. (1992), Induction of finite-state automata using second-order recurrent networks, in J. E. Moody, S. J. Hanson & R. P. Lippmann, eds, '*Advances in Neural Information Processing Systems*', Vol. 4, Morgan Kaufmann Publishers, Inc., pp. 309–317.

Wiles, J. & Elman, J. L. (1995), Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent neural networks, in '*Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*', Cambridge MA: MIT Press, pp. 482–487.

Young, K. & Crutchfield, J. P. (1993), 'Fluctuation spectroscopy', *Chaos, Solutions, and Fractals* **4**, 5–39.

Zeng, Z., Goodman, R. M. & Smyth, P. (1993), 'Learning finite state machines with self-clustering recurrent networks', *Neural Computation* **5**(6), 976–990.

Ziemke, T. & Thieme, M. (2002), 'Neuromodulation of reactive sensorimotor mappings as a short-term memory mechanism in delayed response tasks', *Adaptive Behavior* **10**(3/4), 185–199.