

Semiring Artificial Neural Networks and Weighted Automata

And an Application to Digital Image Encoding

Sebastian Bader¹, Steffen Hölldobler¹, and Alexandre Scalzitti²

¹ International Center for Computational Logic, Technische Universität Dresden

² Institute of Theoretical Computer Science, Department of Computer Science,
Technische Universität Dresden

Abstract. In their seminal paper [1] McCulloch and Pitts have shown the strong relationship between finite automata and so-called McCulloch-Pitts networks. Our goal is to extend this result to weighted automata. In other words, we want to integrate artificial neural networks and weighted automata. For this task, we introduce semiring artificial neural networks, that is, artificial neural networks which implement the addition and the multiplication of semirings. We present a construction of a semiring artificial neural network from a given weighted automaton, and back again. After that, we show how we can approach the problem of encoding an image into a weighted automaton by using a semiring artificial neural network in this process.

1 Introduction

In one of the first and most influential papers in the area of Artificial Neural Networks, McCulloch and Pitts have shown a strong relationship between finite automata and so-called McCulloch-Pitts networks [1] (see also [2]). In this paper we want to extend this correspondence to weighted automata.

Weighted automata are an extension of finite automata which is obtained by assigning costs to the transitions, where the costs are taken from a semiring. These automata are well understood from a declarative point of view, i.e., we know what they do and we know what they are capable of (see e.g. [3, ?]). Fig. 1 shows a simple finite automaton and weighted automaton, which will be explained in more detail in chapter 2.2.

Artificial neural networks are inspired by the information processing performed by animal or human brains. Typically, they are biologically plausible, massively parallel, robust, well-suited to learn and to adapt to new environments, and they degrade gracefully. In a nutshell an artificial neural network consists of a set of simple computational units which are connected to each other (see Fig. 2). Each unit receives (weighted) inputs either externally or from other units, performs a simple operation and propagates the result via all its outgoing connections to the successor units. The connections between the units

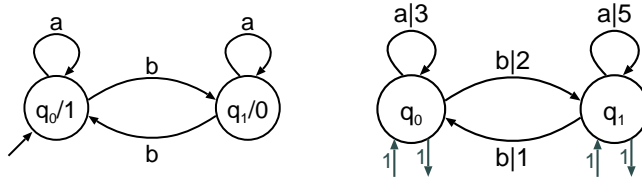


Fig. 1. A simple finite automaton as well as a weighted automaton with states q_0 and q_1 and transitions a and b . In case of a weighted automaton a certain cost is assigned to each transition.

are usually weighted, i.e., the input of a unit is weighted according to the connection via which it is transmitted. A unit can neither distinguish the units from which it receives information, nor the units it sends information to. If a unit does not have any incoming connections it will be called an input unit, since it needs to be activated from outside. A unit without outgoing connections will be called an output unit. By adjusting the weights of the connection the behaviour of the network can be changed; the network is said to be trained. See e.g. [4] for a detailed description of artificial neural networks.

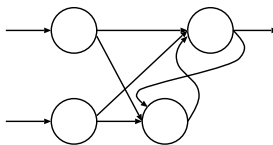


Fig. 2. A simple artificial neural network. Units are interconnected via directed arcs. Weights are omitted.

Artificial neural networks have been successfully applied to many tasks that involve function approximation. Unfortunately we usually have no access to the knowledge encoded in the (weights of the) connections of a given artificial neural network. This implies that we usually lack a declarative description of what is going on in an artificial neural network.

Our goal is to integrate both models of computation, weighted automata and artificial neural networks, in order to obtain a trainable system which has a declarative semantics. In this paper we want to generalize the well known relation between finite automata and McCulloch-Pitts networks [1, 2] by introducing semiring artificial neural networks and showing their correspondence with weighted automata.

Our paper is organized as follows. First we will briefly present the required background on semirings, weighted automata and artificial neural networks. We also define a new extension of artificial neural networks, namely semiring artificial

neural networks. Thereafter, in Section 3 we will show a correspondence between weighted automata and semiring artificial neural networks. In Section 4, we will present an example where we apply our results to image encoding. Finally, in Section 5 we will review our findings and point to future work.

2 Background

2.1 Semirings and Formal Power Series

A *semiring* K is a structure $K = (K, \oplus, \odot, 0_K, 1_K)$ such that

1. $(K, \oplus, 0_K)$ is a commutative monoid, that is, \oplus is a commutative associative binary operation on K and 0_K is the neutral element with respect to \oplus ;
2. $(K, \odot, 1_K)$ is a monoid, that is, \odot is an associative binary operation on K ;
3. \odot is both left and right distributive over \oplus , that is, for all $x, y, z \in K$ it holds that $x \odot (y \oplus z) = x \odot y \oplus x \odot z$ and that $(y \oplus z) \odot x = y \odot x \oplus z \odot x$ and
4. 0_K is absorbing with respect to \odot , that is, for all $x \in K$, it holds that $0_K \odot x = x \odot 0_K = 0_K$.

We call \oplus and \odot respectively the *addition* and the *multiplication* of the semiring K . We call 0_K *neutral element with respect to the addition* and 1_K *neutral element with respect to the multiplication*. Examples for semirings are

1. the boolean semiring $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ with \vee acting as addition and \wedge acting as multiplication;
2. the natural numbers $(\mathbb{N}, +, \cdot, 0, 1)$ with the usual addition and multiplication;
3. the real numbers $(\mathbb{R}, +, \cdot, 0, 1)$ with the usual addition and multiplication;
4. the real max-plus semiring $\mathbb{R}_{\max} = (\mathbb{R}_{\geq 0} \cup \{-\infty\}, \max, +, -\infty, 0)$ with \max acting as addition and the usual addition of real numbers $+$ acting as multiplication. Moreover, $\mathbb{R}_{\geq 0} = [0, \infty)$ with the convention $-\infty + x = -\infty = x + -\infty$ for all $x \in \mathbb{R}_{\max}$;
5. stochastic semiring $([0, 1], \max, \cdot, 0, 1)$ with $[0, 1] \subseteq \mathbb{R}$ with \max acting as addition and the usual multiplication \cdot acting as multiplication;
6. distributive lattices: in this case, \vee is interpreted as the semiring addition and \wedge is interpreted as the semiring multiplication.

Let n be a positive integer. Let us consider the set $M^{n \times n}$ of all matrices of dimension $n \times n$ with entries taken from K . We define the addition \oplus_M and the multiplication \odot_M of $M^{n \times n}$ as follows. Let $A, B \in M^{n \times n}$. For $i, j \in \{1, \dots, n\}$ we define

1. $(a \oplus_M b)_{ij} := (a)_{ij} \oplus (b)_{ij}$ and
2. $(a \odot_M b)_{ij} := \bigoplus_{k=1}^n a_{ik} \odot b_{kj}$.

The neutral element with respect to the addition \oplus_M is the matrix 0_M such that all entries are 0_K and the neutral element with respect to the multiplication

is the matrix 1_M such that its main diagonal has entries 1_K and the rest of 1_M has entries 0_K . We observe that $M = (M^{n \times n}, \oplus_M, \odot_M, 0_M, 1_M)$ is a semiring. We also observe that we can compute, for example, $A_{p \times q} \odot_M B_{q \times r}$, with $1 \leq p, q, r \leq n$.

An *alphabet* is a non-empty finite set. Its elements are called *symbols* or *letters*. In further definitions in this paper we consider a fixed alphabet Σ . A (*finite*) *word* w over Σ is a finite sequence $a_1 a_2 \dots a_n$ of symbols of Σ . We say that n is the *size* of w . If n is 0 then w is the *empty word* and we denote it by ε . We denote the set of words over Σ by Σ^* . We call a function $f : \Sigma^* \rightarrow K$ a *formal power series* with values in the semiring K .

2.2 Weighted Automata

In this section we will introduce an extension of finite automata, by associating weights to the transactions.

Let K be a semiring. A *K-weighted automaton* \mathcal{A} is a tuple (Q, T, c, λ, ρ) such that

1. Q is a non-empty finite set of *states*;
2. a finite set $T \subseteq Q \times \Sigma \times Q$ of *transitions*;
3. $c : T \rightarrow K$ is a function which assigns costs to transitions;
4. $\lambda, \rho : Q \rightarrow K$ are cost functions for *entering* respectively *leaving* each state.

The underlying finite automaton is non-deterministic, i.e. there can be multiple transactions with the same label starting in a certain state, but for two states and one label there can be only one weighted transaction. For the sequel of this paper we will treat c as a function mapping every possible transaction, not only those contained in T to an element of K , by assuming that every non-included transaction will be mapped to 0_K .

Let \mathcal{A} be a K -weighted automaton. A *finite path* P in \mathcal{A} is a finite word over T of the form $(p_i, a_{i+1}, p_{i+1})_{i \in \{0, \dots, n-1\}}$ for some positive integer n . The *label* of P is the finite word $w := a_1 a_2 \dots a_n$. We also say that P is a *w-labeled path* from q_0 to q_n .

Let $P := (p_i, a_{i+1}, p_{i+1})_{i \in \{0, \dots, n-1\}}$ be an arbitrary finite path in \mathcal{A} . The *running cost* of P in \mathcal{A} , denoted by $\text{rcost}_{\mathcal{A}}(P)$, is defined by:

$$\text{rcost}_{\mathcal{A}}(P) := \begin{cases} \odot_{i=0}^{n-1} c(p_i, a_{i+1}, p_{i+1}) & \text{if } n > 0, \\ 1_K & \text{if } n = 0. \end{cases}$$

The *cost* of P , denoted by $\text{cost}_{\mathcal{A}}(P)$, is defined by:

$$\text{cost}_{\mathcal{A}}(P) := \lambda(p_0) \odot \text{rcost}_{\mathcal{A}}(P) \odot \rho(p_n).$$

The *behavior* of \mathcal{A} , denoted by $\|\mathcal{A}\|$, is the function $\|\mathcal{A}\| : \Sigma^* \rightarrow \mathbb{R}_{\max}$ defined by

$$(\|\mathcal{A}\|, w) := \bigoplus \{\text{cost}_{\mathcal{A}}(P) \mid P \text{ is } w\text{-labeled path in } \mathcal{A}\},$$

where $w \in \Sigma^*$. We observe that if the above defined set is empty, then $(\|\mathcal{A}\|, w) := 0_K$.

In the sequel, we present an alternative representation of weighted automata which is very suitable for computations and can be easily implemented by an artificial neural network. Let $\mathcal{A} = (Q, T, c, \lambda, \rho)$ be a K -weighted automaton with n states. We say that \mathcal{A} has a *matrix representation* if there are a row vector $I \in K^{1 \times n}$, a column vector $F \in K^{n \times 1}$, and for each $a \in \Sigma$ a matrix $W_a \in K^{n \times n}$ such that for every $w := a_1 a_2 \dots a_k \in \Sigma^*$,

$$(\|\mathcal{A}\|, w) = I \odot_M W_{a_1} \odot_M W_{a_2} \odot_M \dots \odot_M W_{a_k} \odot_M F.$$

Without going into the details and without showing the detailed proof we state the following lemma

Lemma 1. *For each weighted automaton, as defined above, there exists a matrix representation.*

Proof. To show the existence of this matrix representation we will construct the necessary vectors and matrices. We will consider the states to be ordered (q_0, \dots, q_n) .

1. The initial weight vector I is constructed by applying λ to every state, i.e. $I := (\lambda(q_0), \dots, \lambda(q_n))$
2. The final weight vector F is constructed by applying ρ to every state, i.e. $F := (\rho(q_0), \dots, \rho(q_n))$
3. The weight matrices W_a for every $a \in \Sigma$ have the form:

$$W_a := \begin{pmatrix} c(q_0, a, q_0) & \dots & c(q_0, a, q_n) \\ \vdots & & \vdots \\ c(q_n, a, q_0) & \dots & c(q_n, a, q_n) \end{pmatrix}$$

Using this constructions it is easy to see, that the result is a matrix representation for the given weighted automaton. The details of the proof can be found in [?].

Example 1. We consider the weighted automaton of the Fig. 3. Its weights are taking from the semiring $(\mathbb{R}, +, \cdot, 0, 1)$ that is, the semiring of real numbers with usual addition and multiplication. The alphabet Σ in this case is $\{a, b\}$. We write

$$I := (1 \ 1), \quad W_a := \begin{pmatrix} 3 & 0 \\ 0 & 5 \end{pmatrix}, \quad W_b := \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}, \quad F := \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

2.3 Semiring Artificial Neural Networks

Within this section we want to introduce a new and very general type of artificial neural network. Instead of working on the real numbers only we will lift the mathematics to arbitrary semirings. Let $K = (K, \oplus, \odot, 0_K, 1_K)$ be such a

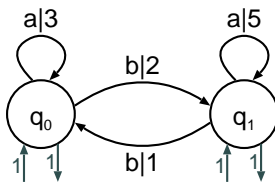


Fig. 3. An \mathbb{R} -weighted automaton with states q_0 and q_1 and transitions a and b .

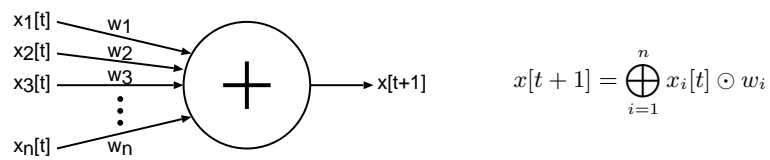


Fig. 4. Schematic plot and dynamics of \oplus -units.

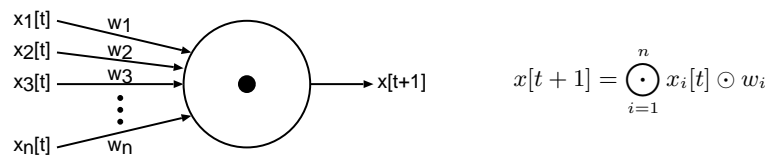


Fig. 5. Schematic plot and dynamics of \odot -unit.

semiring. The weights and activations of the connections and units, respectively, are taken from K . Using the operations provided by the semiring we can easily define two new types of units called \oplus -units and \odot -units and shown in Fig. 4 and Fig. 5, respectively.

The unit shown in Fig. 4 computes a weighted sum of the inputs, where the activation of each unit is \odot -multiplied with the weight of the connection. Thereafter the \oplus -sum of the weighted inputs is computed and propagated to the successor units. Analogously, the unit shown in Fig. 5 computes a weighted product of its inputs.

Definition 1 *Let K be a fixed semiring. Let \mathcal{N} be an artificial neural network consisting of \oplus - and \odot -units, and K -weighted connections. Then we will call \mathcal{N} a semiring artificial neural network.*

One should observe that such a network may be more abstract than the usual artificial neural networks and, depending on the underlying semiring, \oplus - as well as \odot -units may represent whole ensembles of standard units. For example, in case of the natural or real numbers as semirings \oplus -units are simply computing the weighted sum of their inputs with identity as output function, whereas in case of the boolean semiring an \oplus -unit represents a McCulloch-Pitts network consisting of and- and or-units.

Definition 2 *Let \mathcal{N} be a semiring neural network with n designated input neurons and exactly 1 output neuron. Then we will call \mathcal{N} an $n-1$ semiring artificial neural network.*

In the following section we will show, that for each weighted automaton there exists an $n-1$ semiring neural network, such that the input-output behaviour of both systems is equivalent.

3 Constructions

Lemma 2. *Let $\mathcal{A} := (Q, T, c, \lambda, \rho)$ be a K -weighted automaton. Then there exists an $n-1$ semiring neural network \mathcal{N} which simulates \mathcal{A} which means that for every word $w \in \Sigma^*$, \mathcal{N} outputs $(\|\mathcal{A}\|, w)$.*

Proof. To proof the lemma we will give a construction in the sequel of this chapter.

In what follows we will consider the weighted automaton \mathcal{A} to be fixed. We will construct a network \mathcal{N} simulating \mathcal{A} . \mathcal{N} has a very particular topology which consists of 4 layers as described below.

1. Input layer: it consists of σ -labeled units, where $\sigma \in \Sigma$, i.e., for each $\sigma \in \Sigma$ there is exactly one unit. The activation of each σ -labeled unit will be set from outside to 1_K if the current input symbol is σ and 0_K otherwise. For example, to input the word $w := a_1 a_2 \dots a_n \in \Sigma^*$ to \mathcal{N} , in a first step the

- a_1 -labeled unit is set to 1_K and the other input units is set to 0_K ; in a second step a_2 is activated by 1_K and the other input units are set to 0_K , and so on.
2. Gate layer: it consists of \ominus -units. For each pair (q, σ) there is a \ominus -unit t and a connection with weight 1_K is drawn from the σ -unit of the input layer to t .
 3. State layer: it consists of \oplus -units. For each $q \in Q$ there is an \oplus -unit. The gate and the state layer are fully connected with weights set according to the following rule:
 Let u be the \ominus -unit of the gate layer of \mathcal{N} which was associated with the pair (q, σ) and u' be the \oplus -unit associated to $q' \in Q$, in the state layer. The weight of the connection between u and u' is set to $c(q, \sigma, q')$ if $(q, \sigma, q') \in T$, and to 0_K otherwise.
 Moreover, a recurrent connection with weight 1_K from u' to each of the \ominus -nodes of the gate layer labeled (q, σ) .
 4. Output layer: it consists of a single \oplus -unit. For every state $q \in Q$, a connection with weight $\rho(q)$ from the \oplus -unit of the state layer which is associated with q and the \oplus -unit of the output layer is drawn.

The constructed network will consists of $|\Sigma| + |Q| \cdot |\Sigma| + |Q| + 1$ neurons. And, as shown below, it will compute the output for a word of length n in $2n + 3$ time steps, considering the update of a layer as one step.

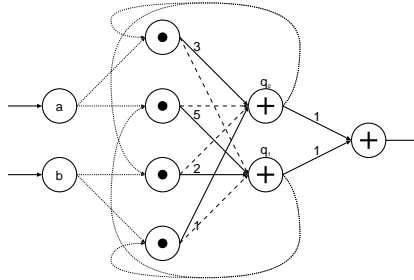


Fig. 6. The semiring artificial neural network corresponding to the weighted automata shown in Fig. 3. It consists (from left to right) of the input, the gate, the state and the output layer. The dotted connections are fixed to 1_K and the dashed ones to 0_K .

Fig. 6 shows the network constructed for the weighted automaton in Fig. 3. This type of architecture is a mixture to the ones known as Elman- [5] and Jordan-networks [6] and possesses a similar dynamics. We have to preset the activations of the state layer units to the corresponding λ -values. To process the empty word we propagate this activation to the output layer, which is depicted in Fig. 7. In case of non-empty words as input-sequence $w = a_0 a_1 a_2 \dots$ we set the activation of the state layer units as above and activate the input-neuron

corresponding to a_0 . These activations are propagated in one step. Thereafter we activate the input-neuron corresponding to a_1 and proceed. Every second time-step $t = 2n$ we activate the neuron of the input layer, which corresponds to a_n . At time $(2n) + 3$ the activation of the output layer corresponds to the output of the automaton after processing $a_0a_1 \dots a_n$.

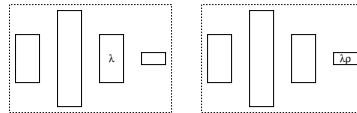


Fig. 7. The schema of a semiring artificial neural network processing the empty word.

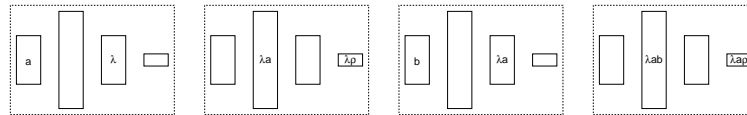


Fig. 8. The schema of a semiring artificial neural network processing a non-empty word.

During the training process of the network the weights between gate and state layer as well as between state and output layer, and the initial activation of the state layer are adjusted. For certain semirings, e.g. $(\mathbb{R}, +, \cdot, 0, 1)$, this can be done using simple back-propagation through time as described in [5, 7]. In Section 4 we will give an example of this process. Moreover, we can construct a network of this topology given an alphabet Σ and a number of states without an automaton by simply assigning random values to the weights between gate and state layer as well as state and output layer.

Proposition 3 *For each network of the type described above there exists a weighted automaton, such that both systems behave equivalently, i.e., for every word $w \in \Sigma^*$ we get the same output.*

Proof. The corresponding automaton can easily be extracted from the weight matrix between gate and state layer, which encodes the transitions of the automaton.

Other architectures are possible as well, but this one was chosen for several reasons. As mentioned above it is similar to well known architectures as Jordan- and Elman-networks. Furthermore, all recurrent connections have an associated weight of 1_K , which simplifies the training process.

4 An Example - Digital Images

In this section we want to discuss the problem of image encoding by weighted automata, that is, the problem of constructing a weighted automaton which computes a given image. Firstly we define images. An *image* of resolution $r \times s$ for our purposes is a matrix of r rows and s columns of *pixels*. We can assign to each pixel a *gray-scale intensity*. Without loss of generality, we will consider images of resolution $2^n \times 2^n$ only, and for pixels we take gray-scale intensities in the interval $[0, 1]$. Culik and Kari in [8] presented a first algorithm that constructs a weighted automaton which encodes a given image. The problem is that we cannot fix the number of states of this weighted automaton in advance. Culik and Kari in [9] and, later on, Katritzke in [10] proposed improvements of this algorithm which compute approximations of the image and terminate. In this section we want to use the results obtained in the last section to construct a weighted automaton by means of training a semiring artificial neural network.

But first, we need to introduce a method to address pixels of a given image. Let n be a natural number and let us consider an image of resolution $2^n \times 2^n$. We assign to each pixel a word of length n over the alphabet $\{0, 1, 2, 3\}$ as follows.

1. If $n = 0$ then the image contains exactly one pixel which is addressed by the empty word ε .
2. If $n = 1$ then the image contains 2×2 pixels which can be addressed as shown in the left-hand side of Fig. 9.
3. If $n = 2$ then the addresses are shown in the right-hand side of Fig. 9.
4. The addresses for $n > 2$ are obtained inductively.

1	3
0	2

11	13	31	33
10	12	30	32
01	03	21	23
00	02	20	22

Fig. 9. Pixel addressing for $n = 1$ (left-hand side) and $n = 2$ (right-hand side).

Roughly speaking, the pixel addressing method presented above provides us a set of finite words which can be interpreted as follows. Let w be a pixel address. If $w = v \cdot a$ with $a \in \Sigma$, then w addresses quadrant a of the sub-image addressed

by v . To each address w we assign the average gray-scale c of the sub-image addressed by w , thereby we obtain a function $f : \Sigma^* \rightarrow [0, 1]$.

Let \mathcal{A} be an \mathbb{R} -weighted automaton and let $f_{\mathcal{A}}$ be its behavior. We decode the image $f_{\mathcal{A}}$ at resolution $2^k \times 2^k$ by computing $f_{\mathcal{A}}(w)$ for every $w \in \Sigma^*$ of length k .

We want to construct an automaton whose behaviour is f , or at least similar to f . To achieve this goal we constructed a randomly initialized semiring artificial neural network as described above. In addition, we extracted a set of training samples from a given image and used this samples to train the network using back-propagation through time, as described in [5]. Fig. 10 shows a sample, the network output prior learning and the output after learning, each shown as a picture.

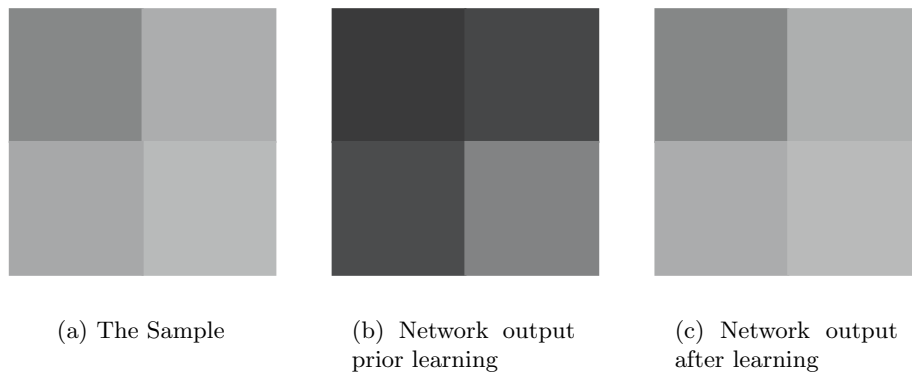


Fig. 10. Sample and Network Output.

5 Conclusions and Future Work

In this paper we have introduced semiring artificial neural networks. These networks can be seen as a generalization of the so-called $\Sigma - \Pi$ -networks where the values as well as addition and multiplication are taken from an arbitrarily given semiring. We have shown that for each weighted automaton there exists such a network with identical input-output behaviour. We have thus extended the work by McCulloch and Pitts [1, 2]. In addition, we have shown by example of image encoding that back-propagation through time can be applied to train semiring artificial neural networks.

In section 3 we have presented a construction method which receives as input a weighted automaton and outputs a $n-1$ -semiring neural network of a particular topology. We can also do the inverse: given a $n-1$ -semiring neural network of the particular topology as the outputs of the above mentioned construction method

we can construct a weighted automaton with same behaviour. The conversion of arbitrary networks is still an open problem.

Let us review this application: The first algorithm of Culik and Kari in [8] receives as input an (possibly infinite) image and terminates if and only if the image is average preserving. This is a property which can be formulated in terms of linear dependence of a certain set of functions which can be constructed from the image. If the algorithm terminates, we have as output a weighted automaton which represents or encodes exactly the given image. Moreover, the number of states is exactly the dimension of the vector space whose basis is this set of functions mentioned above.

Depending on the number of states we may be interested in a weighted automaton with less states and which encodes an approximation of the given image. In our approach, we can construct a semiring artificial neural network with a fixed number of states and with the topology as discussed in Section 3. This network can be trained using Back-Propagation, and thereafter, can be translated to a weighted automaton.

Taking a closer look at the transaction structure created by Kari's algorithm, we find that there are no loops of transactions, which might restrict the automaton. Please note that this is not true for the automata constructed using our approach. The consequences, regarding the expressive power need further investigations.

Culik and Kari [8] and later on Katritzke [10] made several improvements to the algorithm which we considered above. Particularly Katritzke implemented several algorithms concerning image encoding and compression in his software AutoPic. We intend to do a complexity comparison between AutoPic and our approach, and furthermore, extensive tests of the dynamics of the networks, including the training process.

References

- [1] McCulloch, W.S., Pitts, W.: A logical calculus and the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5** (1943) 115–133
- [2] Arbib, M.A.: *Brains, Machines, and Mathematics*. 2nd edn. Springer (1987)
- [3] Schützenberger, M.P.: On the definition of a family of automata. *Information and Control* **4** (1961) 245–270
- [4] Hertz, J., Krogh, A., Palmer, R.G.: *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company (1991)
- [5] Elman, J.L.: Finding structure in time. *Cognitive Science* **14** (1990) 179–211
- [6] Jordan, M.I.: Attractor dynamics and parallelism in a connectionist sequential machine. (1986) 531–546
- [7] Werbos, P.J.: Backpropagation through time: What it does and how to do it. In: *Proceedings of the IEEE*. Volume 78. (1990) 1550–1560
- [8] Culik, K., Kari, J.: Image-data compression using edge-optimizing algorithm for wfa inference. *Journal of Information Processing and Management* **30** (1994) 829–838
- [9] Culik, II, K., Kari, J.: Digital images and formal languages. In: *Handbook of formal languages*, Vol. 3. Springer, Berlin (1997) 599–616

- [10] Katritzke, F.: Refinements of data compression using weighted finite automata. PhD thesis, Universität Siegen, Germany (2002)