June, 2001

Verifying Properties of Neural Networks

Pedro Rodriques
J. Félix Costa
Hava Siegelmann, *University of Massachusetts - Amherst*

Verifying Properties of Neural Networks

Pedro Rodrigues[1], J. Félix Costa[2], and Hava T. Siegelmann[3]

[1] Departamento de Informática, Faculdade de Ciências da Universidade de Lisboa, Campo Grande, 1749-016 Lisboa, Portugal
pmr@di.fc.ul.pt
[2] Departamento de Matemática, Instituto Superior Técnico, Lisbon University of Technology, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
fgc@math.ist.utl.pt
[3] Faculty of Industrial Engineering and Management, Technion City, Haifa 32 000, Israel
iehava@ie.technion.ac.il

Abstract. In the beginning of nineties, Hava Siegelmann proposed a new computational model, the Artificial Recurrent Neural Network (ARNN), and proved that it could perform hypercomputation. She also established the equivalence between the ARNN and other analog systems that support hypercomputation, launching the foundations of an alternative computational theory. In this paper we contribute to this alternative theory by exploring the use of formal methods in the verification of temporal properties of ARNNs. Based on the work of Bradfield in verification of temporal properties of infinite systems, we simplify his tableau system, keeping its expressive power, and show that it is suitable to the verification of temporal properties of ARNNs.

1 Introduction

During the forties and the beginning of the fifties, the Josiah Macy Jr. Foundation patronized an annual scientific meeting that brought together researchers from disparate areas such as Biology, Physics, Mathematics and Engineering, to discuss possible models of computation machines. They all agreed that a computer should resemble the brain. From this principle, two research programs emerged: the first one, less known, originated the theory and technology of analog computation; the second one, focused on the theory and technology of digital computation.

In 1943, McCulloch and Pitts presented their most influential paper "A logical calculus of the ideas immanent in nervous activity", in which they introduced the idea that processes in the brain can be metaphorically modeled by a finite interconnection of logical devices. Although the logical abstraction of McCulloch and Pitts was controversial, von Neumann developed this idea towards the design of the digital computer and, since then, the digital approach to computation has prevailed.

Recently, researchers have speculated that although the Turing machine, the theoretical model of digital computation, is indeed able to simulate a large class of computations, it does not necessarily provide a complete picture of the computations possible in nature. This speculation emerged with several proposals of different models of computation. In the beginning of nineties, Hava Siegelmann proposes what

she called the "Artificial Recurrent Neural Network" (ARNN), an analog model capable of performing hypercomputation. By showing that ARNNs were equivalent to other analog computational models, also able to perform hypercomputation, Siegelmann claims that ARNNs should be taken as *the* analog computational model. Upon this postulate some research has been conducted: foundations [11, 9, 5], languages and compilers [6, 10, 7, 8], etc.
In this paper we will be contributing to this emergent alternative computer science by exploring the use of formal methods in the verification of temporal properties of ARNNs. More specifically, we will be using "model checking". "Model checking" is a way of certifying the agreement between a specification and a system: a system is described as a transition graph and the proof of properties is established by traversing parts of the graph. We will extend the work of Bradfield [1, 2, 3] who has developed a "mechanism" of "model checking" for infinite systems and analyzed its applicability to conventional Petri Nets, by simplifying the "mechanism" and studying its applicability to ARNNs.
In section 2 we will present the language we shall be using to describe properties we wish to verify. In section 3, we will present the tableau system and, in section 4, we will present the ARNN and the applicability of the tableau system to prove properties of ARNNs.

2 Mu-calculus (Syntax and Semantics)

Mu-calculus is a temporal logic with vast expressive power – it subsumes several standard temporal logics [4]. There are several versions of it in the literature. We shall be using the version that Bradfield used in his PhD thesis [1], a version in which the indexation of modal operators was enlarged from labels to sets of labels.
Let us assume two fixed sets: the set of variables $\mathcal{V}$ (countably infinite) and the set of labels $\mathcal{E}$ (finite and non-empty). X, Y and Z will be used to designate variables and, a,b will be used to designate labels.

Definition A mu-formula is inductively defined as follows:
- X is a mu-formula;
- if φ_1 and φ_2 are mu-formulas, then $(\varphi_1 \wedge \varphi_2)$ and $(\varphi_1 \vee \varphi_2)$ are mu-formulas;
- if φ is a mu-formula, then $(\neg\varphi)$, $([K]\varphi)$ and $(\langle K\rangle\varphi)$ are mu-formulas, where K stands for a set of labels;
- if φ is a mu-formula and any free occurrence of X in φ is within the scope of an even number of negations, then $(\nu X.\varphi)$ and $(\mu X.\varphi)$ are mu-formulas. ◊

Notation To minimize the use of parenthesis, we adopt the following convention of precedence (from greater to a smaller precedence): $\neg$, the modal operators [K] and $\langle K\rangle$, the logical operators $\wedge$ and $\vee$, and, finally, the fix-point operators $\nu X.\varphi$ and $\mu X.\varphi$. We also assume that the logical operators associate to the left. ◊

Notation We will be using $\sigma X.\varphi$ to denote either of the two fix-point operators. ◊

Mu-formulas are interpreted in models.

Definition A labeled transition system is a triple $\langle S, Act, \rightarrow \rangle$ where:
- S is a set whose elements are called states;
- Act is a set whose elements are called actions;
- $\rightarrow \subseteq S \times Act \times S$ is a relation, called the transition relation. ◊

Definition A model M for the set of mu-formulas is a pair $\langle T, I \rangle$ where $T = \langle S, E, \rightarrow \rangle$ is a labeled transition system, with E as the action set, and $I: V \rightarrow 2^S$, an interpretation that assigns a set of states to each variable. ◊

Definition [1] Let $M = \langle T, I \rangle$ be a model for the set of mu-formulas. The denotation $\|\varphi\|_I^T$ of a mu-formula in a model M is inductively defined as follows:
- $\|X\|_I^T = I(X)$
- $\|\neg\varphi\|_I^T = S - \|\varphi\|_I^T$
- $\|\varphi_1 \wedge \varphi_2\|_I^T = \|\varphi_1\|_I^T \cap \|\varphi_2\|_I^T$
- $\|\varphi_1 \vee \varphi_2\|_I^T = \|\varphi_1\|_I^T \cup \|\varphi_2\|_I^T$
- $\|[K]\varphi\|_I^T = \{ s \in S : \forall s' \in S \; \forall a \in K \; (s \xrightarrow{a} s' \Rightarrow s' \in \|\varphi\|_I^T) \}$
- $\|\langle K \rangle \varphi\|_I^T = \{ s \in S : \exists s' \in S \; \exists a \in K \; (s \xrightarrow{a} s' \; e \; s' \in \|\varphi\|_I^T) \}$
- $\|\nu X.\varphi\|_I^T = \cup \{ T \subseteq S : T \subseteq \|\varphi\|_{I[X:=T]}^T \}$
- $\|\mu X.\varphi\|_I^T = \cap \{ T \subseteq S : \|\varphi\|_{I[X:=T]}^T \subseteq T \}$

  where $I[X:=T]$ agrees with I in every variable except, eventually, in X, being $I[X:=T](X) = T$. ◊

Notation We shall be writing $\|\varphi\|_I$ instead of $\|\varphi\|_I^T$ whenever T is irrelevant. ◊

The meaning of mu-formulas not involving fix-point operators should be easily understandable from this last definition. For the meaning of more complex mu-formulas cf. [1, 2].

3 The Tableau System

Bradfield [1] presented a tableau system for proving that a set of states satisfies a property described as a mu-formula without determining its full denotation. We shall be using an equivalent tableau system in which the unfolding rule can only be applied once to each subformula whose external operator is a fix-point operator. This small change had already been mentioned by Bradfield [1] and used by him [2] though not considering the indexation of modal operators by sets of states. In our opinion, this small change simplifies tableaux construction.

We shall not present the proofs of correctness and completeness of the tableau system, but they are fairly similar to the ones presented by Bradfield [1].
Let us assume a fixed model $M = \langle T, I \rangle$. We wish to prove that a set of states S satisfies a property φ, i.e., $S \subseteq \|\varphi\|_I$. This is only possible, using the tableau system, if φ is in the positive normal form: it is proven that every mu-formula has an equivalent mu-formula, which is in the positive normal form (negations are only applied to variables, there are no variables simultaneously free and bounded in the formula and the same variable can not be bounded by two fix-point operators).
A tableau is a proof tree built on sequents of the form $S' \vdash \varphi'$; a sequent represents the goal of showing $S' \subseteq \|\varphi'\|_{I'}$. If we wish to prove that $S \subseteq \|\varphi\|_I$, we start from the sequent $S \vdash \varphi$ and go on applying the rules of the system, that transform goals into one or two subgoals, until we reach terminal sequents. Once built the tableau, the root sequent goal is proven iff a given success condition is verified.

Definition tableau rules (presented as inverted proof rules):

(and) $\dfrac{S \vdash \varphi_1 \wedge \varphi_2}{S \vdash \varphi_1 \quad S \vdash \varphi_2}$

(or) $\dfrac{S \vdash \varphi_1 \vee \varphi_2}{S_1 \vdash \varphi_1 \quad S_2 \vdash \varphi_2}$ where $S = S_1 \cup S_2$

(box) $\dfrac{S \vdash [K]\varphi}{S' \vdash \varphi}$ where $S' = \{ s' \in S : \exists s \in S \, \exists a \in K \, (s \xrightarrow{a} s')\}$

(diamond) $\dfrac{S \vdash \langle K \rangle \varphi}{S' \vdash \varphi}$ where S' is the range of an application $f: S \to S$ such that, if $f(s) = s'$, then $\exists a \in K \, (s \xrightarrow{a} s')$

(unfolding) $\dfrac{S \vdash \sigma X.\varphi}{S \vdash \varphi}$ (weakening) $\dfrac{S \vdash \varphi}{S' \vdash \varphi}$ where $S' \supseteq S$ ◊

These rules, except for the last two, should be, straightforward. To explain the unfolding rule, let us suppose that the goal represented by $S \vdash \sigma X.\varphi$ is $S \subseteq \|\sigma X.\varphi\|_{I'}$. Then, the goal represented by $S \vdash \varphi$ should be seen as $S \subseteq \|\varphi\|_{I'[X:=S]}$. According to the way the semantics is defined, if this latest inclusion is true, then $S \subseteq \|\sigma X.\varphi\|_{I'}$ as long as $\sigma X.\varphi$ is $\nu X.\varphi$. If this is not the case, then something more has to be proven.
The weakening rule is strictly needed for the completeness of the system. Still, it has only to be applied immediately before the unfolding rule. We shall present an example of this later, in the next section.
Given a root sequent, the tableau is built downwards, using tableau rules, until the only rule applicable to leaf sequents is the weakening rule (these sequents are called terminal nodes).

Definition Let φ be the mu-formula present in the root sequent and n' = S' |– φ', a terminal node.

- If φ' is X and X is free in φ, then n' is called a free terminal.
- If φ' is ¬X and X is free in φ, then n' is called a negated terminal.
- If φ' is X and X is bounded in φ, then n' is called a σ-terminal.
- If φ' is X and there is a subformula μX.ψ of φ, then n' is called a μ-terminal.
- If φ' is X and there is a subformula νX.ψ of φ, then n' is called a ν-terminal. ◊

For a tableau to be successful, every terminal node must be a successful node.

Definition Let n' = S' |– X be a σ-terminal. Its companion is the lowest node n''= S'' |– σX.φ above it. ◊

Definition A terminal node n' = S' |– φ' is successful iff:

- φ' is ¬X and S' ∩ I(X) = ∅ or
- S' = ∅ or
- n' is a free terminal and S' ⊆ I(φ') or
- n' is a ν-terminal with companion n'' = S'' |– νX.φ'' and S' ⊆ S'' or
- n' is a μ-terminal with companion n'' = S'' |– μX.φ'', S' ⊆ S'' and n'' satisfies the μ-success condition presented below. ◊

Definition A path from a state s at a node n to a state s' at a node n', s@n $\longrightarrow\!\!\!\!\!\cdot\;$ s'@n', is a sequence (s,n) = (s_0,n_0), (s_1,n_1), …, (s_k,n_k) = (s', n') of pairs (state, node) such that:

- n_{i+1} is a child of n_i;
- if n_i = S_i |– φ_i, then $s_i \in S_i$;
- if the rule applied to n_i is the box or the diamond rule, then ∃a ∈ K ($s_i \xrightarrow{a} s_{i+1}$), otherwise, s_{i+1} é s_i. ◊

Definition There is an extended path from a state s at a node n to a state s' at a node n', s@n $\longrightarrow\!\!\!\!\!\vdots\;$ s'@n', if:

i) s@n $\longrightarrow\!\!\!\!\!\cdot\;$ s'@n' or

ii) there exists a node n'' = S'' |– σX.φ, companion of k (k≥1) σ-terminal nodes, n_1, …, n_k, and a finite sequence of states, s_0, …, s_k, such that s@n $\longrightarrow\!\!\!\!\!\cdot\;$ s_0@n'', for each i (0≤i<k), s_i@n'' $\longrightarrow\!\!\!\!\!\vdots\;$ s_{i+1}@n_{i+1}, and s_k@n'' $\longrightarrow\!\!\!\!\!\vdots\;$ s'@n'. ◊

Definition Let n = S |– σX.φ be a companion node. We define the relation $\sqsupset_n$ on the elements of S in the following way: s $\sqsupset_n$ s' iff s@n $\longrightarrow\!\!\!\!\!\vdots\;$ s'@n' for some σ-terminal n' = S' |– X. ◊

Definition Let n = S |– μX.φ be a companion node. n satisfies the μ-success condition iff the relation $\sqsupset_n$ is well founded. ◊

We will postpone an example of the use of the tableau system to the next section.

4 Tableau System for Neural Networks

The neural network architecture we will be using is the one used by Siegelmann [11] to study the computational power of neural networks. Nevertheless, this model is pretty similar to other neural network architectures and so the work we will present is easily adaptable to those architectures.

Definition An artificial recurrent neural network of n ($n \in \mathbb{N}$) neurons and u ($u \in \mathbb{N}$) input units is a quadruple $\Re_{n,u} = \langle A, B, C, f \rangle$ where: A is a $n \times n$ matrix of real numbers; B is a $n \times u$ matrix of real numbers; C is a $n \times 1$ matrix of real numbers; f is an application from $\mathbb{R}$ to $\mathbb{R}$. ◊

Definition The dynamics of an ARNN $\Re_{n,u} = \langle A, B, C, f \rangle$ is given by an application $F: \mathbb{R}^n \times \{0,1\}^u \to \mathbb{R}^n$ defined in the following way:

$F(x, u) = x^+$ where, for all i such that $1 \leq i \leq n$, $x_i^+ = f\left(\sum_{j=1}^{n} a_{ij}x_j + \sum_{j=1}^{u} b_{ij}u_j + c_i\right)$ ◊

The computation of an ARNN is infinite: given an initial state and an infinite succession of stimulus to be presented to the input units, the net will be evolving through a succession of states infinitely.

Definition Let $\Re_{n,u} = \langle A, B, C, f \rangle$ be an ARNN. An input stream for $\Re_{n,u}$ is an application $u: \mathbb{N}_0 \to \{0,1\}^u$. ◊

Definition Let $\Re_{n,u} = \langle A, B, C, f \rangle$ be an ARNN. A computation of $\Re_{n,u}$ is a triple $\langle x_{init}, u, x \rangle$ where: x_{init} is an element of $\mathbb{R}^n$, the initial state; u is an input stream for $\Re_{n,u}$; x is an application from $\mathbb{N}_0$ to $\mathbb{R}^n$ defined in the following way: $x(0) = x_{init}$; for all $t \geq 0$, $x(t+1) = F(x(t), u(t))$ ◊

As we can notice, an ARNN is not a labeled transition system. Nevertheless, its behavior can be simulated by a labeled transition system.

Definition Let $\Re_{n,u}$ be an ARNN. The associated labeled transition system to $\Re_{n,u}$ is a triple $T_{\Re_{n,u}} = \langle S, Act, \to \rangle$ where $S = \mathbb{R}^n$, $Act = \{0,1\}^u$ and $\forall s, s' \in S \ \forall a \in Act$ ($s \xrightarrow{a} s'$ iff $F(s,a) = s'$). ◊

According to this latest definition, an ARNN and its associated labeled transition system evolve in the same way (a stimulus presented to the input units becomes an action performed on the associated labeled transition system). Therefore, we will say

that a set of states S, of an ARNN $\Re_{n,u}$, satisfy a given property φ iff the same set S is in the denotation of φ, considering $T_{\Re_{n,u}}$ as the labeled transition system.

We will finish this section with an example. Assume an ARNN with 2 neurons and 1 input unit defined as follows:

$A = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $C = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and f is the signal function, i.e., $f(x) = \begin{cases} 0 & \text{se } x \leq 0 \\ 1 & \text{se } x > 0 \end{cases}$

Let us prove that, for all states of the ARNN in which both neurons have the same activation, there is a computation in which the state (1,0), the state in which the neuron 1 has activation 1 and the neuron 2 has activation 0, is visited infinitely often.

Translating our purpose to the mu-calculus, we wish to prove that the set $\{(x,x):x\in\mathbb{R}\}$ is in the denotation of $\nu Y.\mu X.(P \vee \langle - \rangle X) \wedge \langle - \rangle Y$, assuming {(1,0)} as the interpretation for P (we are using $\langle - \rangle$ as an abbreviation for $\langle \{0,1\} \rangle$).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{(x,x):x\in\mathbb{R}\} \vdash \nu Y.\mu X.(P \vee \langle - \rangle X) \wedge \langle - \rangle Y}}{n_1\ \{(x,x):x\in\mathbb{R}\}\cup\{(0,1),(1,0)\} \vdash \nu Y.\mu X.(P \vee \langle - \rangle X) \wedge \langle - \rangle Y}}{n_2\ \{(x,x):x\in\mathbb{R}\}\cup\{(0,1),(1,0)\} \vdash \mu X.(P \vee \langle - \rangle X) \wedge \langle - \rangle Y}}{\{(x,x):x\in\mathbb{R}\}\cup\{(0,1),(1,0)\} \vdash (P \vee \langle - \rangle X) \wedge \langle - \rangle Y}}{\dfrac{\{(x,x):x\in\mathbb{R}\}\cup\{(0,1),(1,0)\} \vdash P \vee \langle - \rangle X}{n_4\{(1,0)\} \vdash P}} \qquad \dfrac{\dfrac{\{(x,x):x\in\mathbb{R}\}\cup\{(0,1),(1,0)\} \vdash \langle - \rangle Y}{n_5\quad \{(0,1),(1,0)\} \vdash Y}}{n_3}$$

$$\dfrac{\{(x,x):x\in\mathbb{R}\}\cup\{(0,1)\} \vdash \langle - \rangle X}{n_7\ \{(0,1),(1,0)\} \vdash X}\ n_6$$

As it can be seen, in this case, it would be impossible to prove what we wanted if the weakening rule (the first rule applied) was not available. This rule is strictly essential to satisfy the inclusion condition of the sets associated to σ-terminals and its companions.

Before showing that this is a successful tableau, we must present the functions implicit in the application of the diamond rule:

- Let f_3 denote the function considered in the application of the diamond rule to n_3. This function is defined in the following way: for all $x \leq 0$, $f_3(x,x) = (1,0)$; for all $x > 0$, $f_3(x,x) = (0,1)$; $f_3(0,1) = (1,0)$, $f_3(1,0) = (0,1)$.
- Let f_5 denote the function considered in the application of the diamond rule to n_5. This function is defined in the following way: for all $x \leq 0$, $f_5(x,x) = (1,0)$; for all $x > 0$, $f_5(x,x) = (0,1)$; $f_5(0,1) = (1,0)$.

Let us prove that the tableau is a successful tableau. n_4 and n_6 are clearly successful terminals: n_4 is a successful terminal because {(1,0)} is contained in the interpretation of P; n_6 is also a successful node because {(0,1), (1,0)} is contained in $\{(x,x):x\in\mathbb{R}\}\cup\{(0,1),(1,0)\}$, the set associated to n_1, the companion of n_6. Finally, n_7 is also a successful terminal since {(0,1), (1,0)} is contained in $\{(x,x):x\in\mathbb{R}\}\cup$ {(0,1), (1,0)}, the set associated to n_2, the companion of n_7, and the relation $\sqsupset_{n_2}$ is well founded: $(0,1) \sqsupset_{n_2} (1,0)$; for all $x \leq 0$ $(x,x) \sqsupset_{n_2} (1,0)$; for all $x > 0$ $(x,x) \sqsupset_{n_2} (0,1)$. Since all terminal nodes are successful, the tableau is a successful tableau.

5 Conclusions

We have presented a system for reasoning about ARNNs by adapting standard techniques of model-checking whose applicability had only been studied on conventional Petri Nets. The system presented is clearly undecidable and so, can not be fully automated. Still, if we confine the elements of the matrices to integers, the ARNN becomes a finite system and, consequently, the tableau system becomes decidable (we have implemented a fully automated tableau system for this subclass of ARNNs). Nevertheless, since we are dealing with a local model-checking technique, we shall be able to automatically prove properties about ARNNs with an infinite state space. How can it be done? We don't have the answer yet and the work on finite ARNNs has not given us much clues. We believe we should recover the work on finite ARNNs and try to improve the results obtained.

References

1. Bradfield, J.: Verifying Temporal Properties of Systems with Applications to Petri Nets, PhD thesis, University of Edinburgh (1991)
2. Bradfield, J.: Proving Temporal Properties of Petri Nets. In Rozenberg, G. (eds.): Advances in Petri Nets 1991. Lecture Notes in Computer Science, Vol. 524. Springer-Verlag, Berlin (1991) 29-47
3. Bradfield, J., Stirling, C.:,Local Model Checking for Infinite State Spaces. Theoretical Computer Science, Vol. 96. (1992) 157-174
4. Emerson, E.A., Lei, C.-L.: Efficient Model Checking in Fragments of the Propositional Mu-calculus. Proceedings of the 1st IEEE Symposium on Logic in Computer Science, (1986) 267-278
5. Gilles, C., Miller, C., Chen, D., Chen, H., Sun, G., Lee, Y.: Learning and Extracting Finite State Automata with Second-Order Recurrent Neural Networks. Neural Computation, Vol. 4 **3** (1992) 393-405
6. Gruau, F., Ratajszczak, J., Wiber, G.: A neural compiler. Theoretical Computer Science, Vol. 141(1-2) (1995) 1-52
7. Neto, J. P., Siegelmann, H. T., Costa, J. F.: On the Implementation of Programming Languages with Neural Nets. In First International Conference on Computing Anticipatory Systems (CASYS'97). CHAOS, **1** (1998) 201-208
8. Neto, J. P., Siegelmann, H. T., Costa, J. F.: Building Neural Net Software. submitted, (1999)
9. Pollack, J.: On Connectionism Models of Natural Language Processing. PhD thesis. University of Illinois, Urbana (1987)
10. Siegelmann, H. T.: On NIL: the Software Constructor of Neural Networks. Parallel Processing Letters. Vol. 6 **4**, World Scientific Publishing Company (1996) 575-582
11. Siegelmann, H. T.: Neural Networks and Analog Computation: beyond the Turing limit. Birkhäuser, Boston (1999)