# Evolving Recurrent Neural Networks are Super-Turing

Jérémie Cabessa and Hava T. Siegelmann

University of Massachusetts Amherst

3 August 2011

# Introduction

- *Architectural Evolution* is a crucial aspect of biological neural networks: synaptic plasticity, cell birth and death, ...
- We studied the computational capabilities of a rate model of *evolving* recurrent neural networks.
- *Evolving* recurrent neural networks are super-Turing.

# Introduction

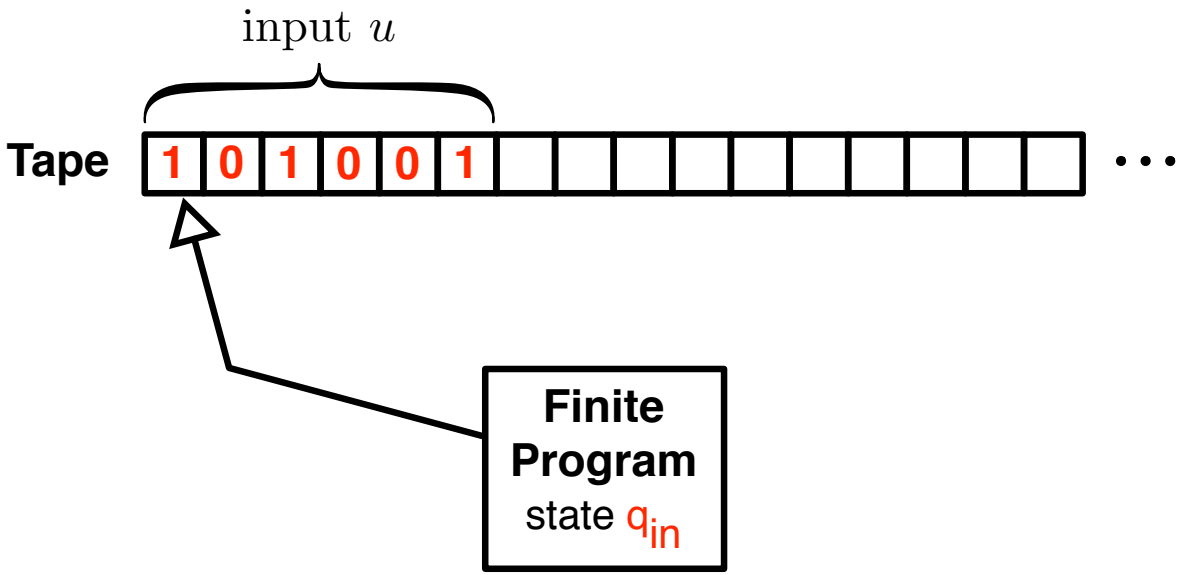
- *Architectural Evolution* is a crucial aspect of biological neural networks: synaptic plasticity, cell birth and death, ...
- We studied the computational capabilities of a rate model of *evolving* recurrent neural networks.
- *Evolving* recurrent neural networks are super-Turing.

# Introduction

- *Architectural Evolution* is a crucial aspect of biological neural networks: synaptic plasticity, cell birth and death, ...
- We studied the computational capabilities of a rate model of *evolving* recurrent neural networks.
- *Evolving* recurrent neural networks are super-Turing.

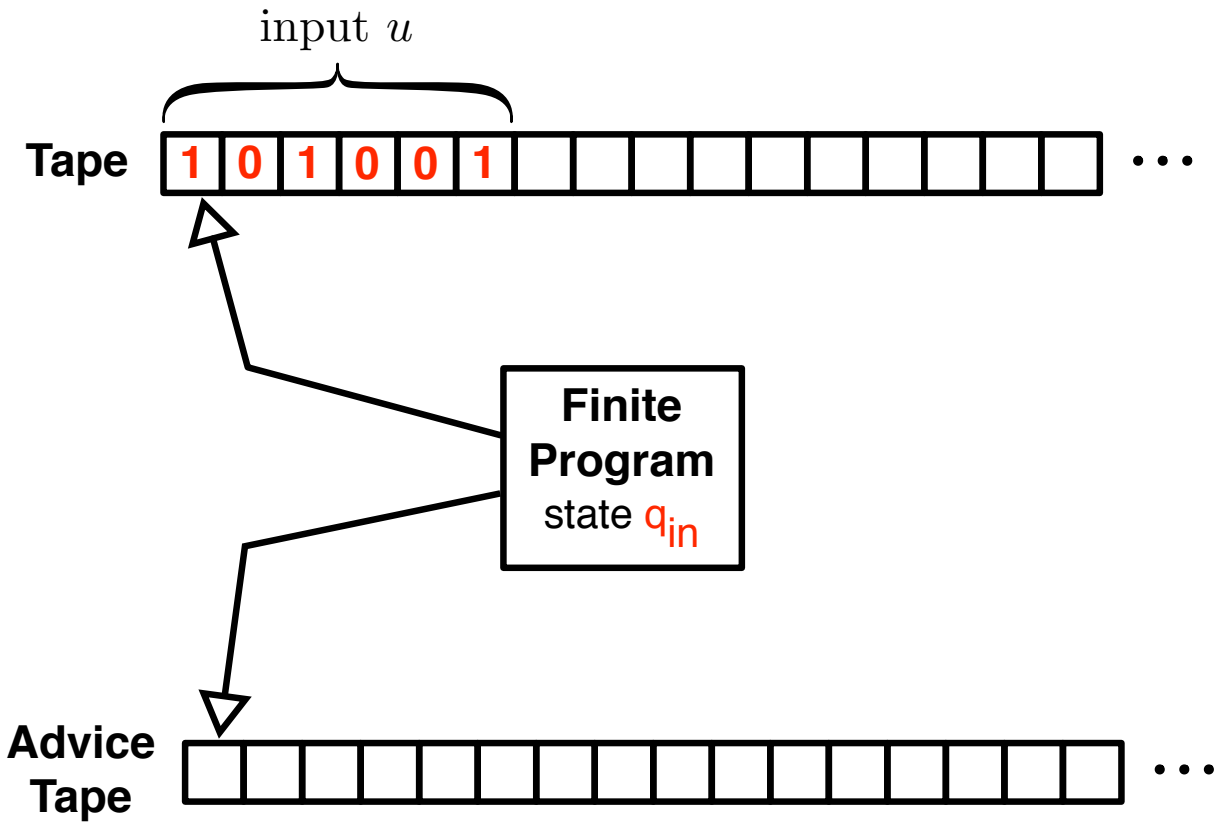
# Turing machines and Turing machines with advice

Turing machine with advice: A Turing machine provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$



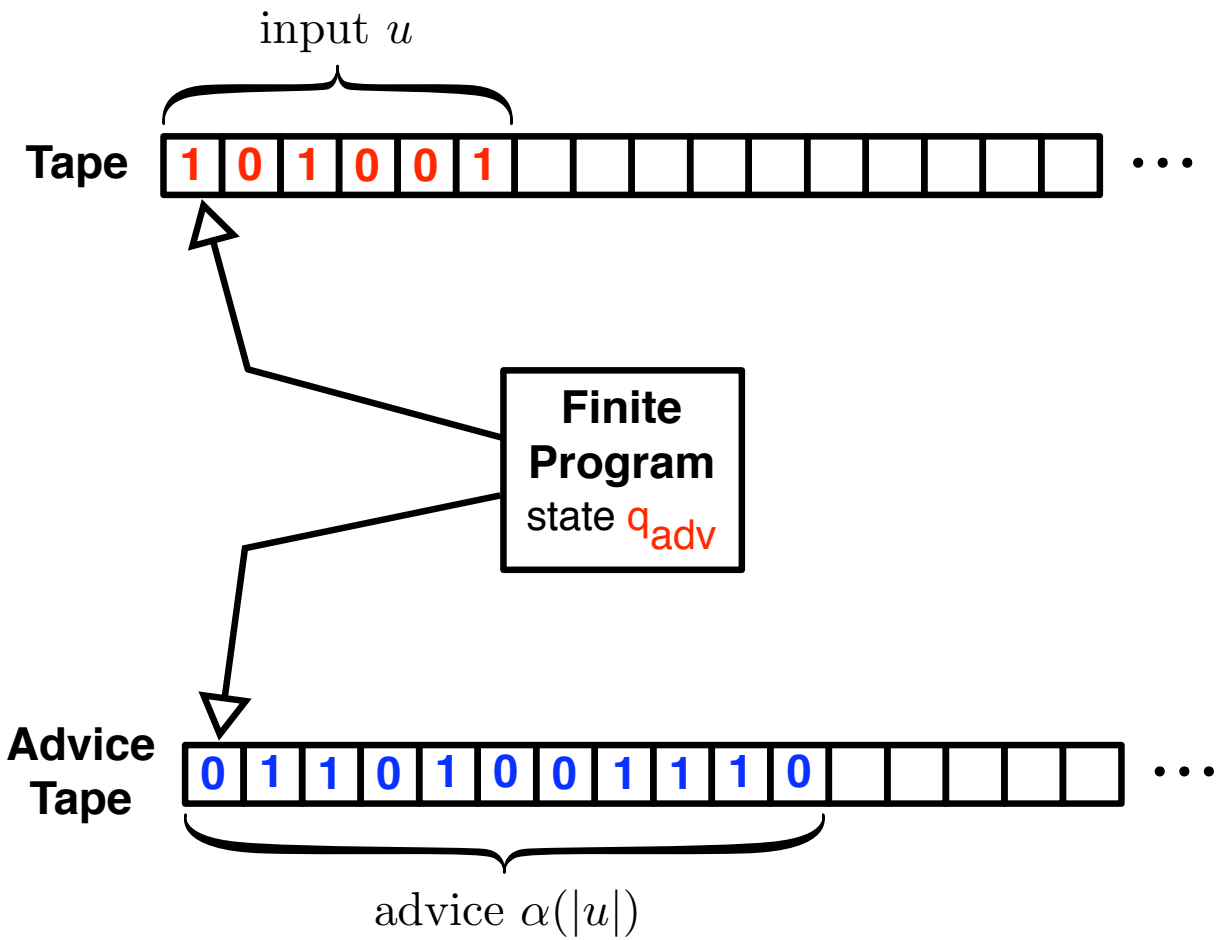
# Turing machines and Turing machines with advice

Turing machine with advice: A Turing machine provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$



# Turing machines and Turing machines with advice

Turing machine with advice: A Turing machine provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$



# TM vs TM/A

We consider Turing machines with polynomially long advice (TM/poly(A)).

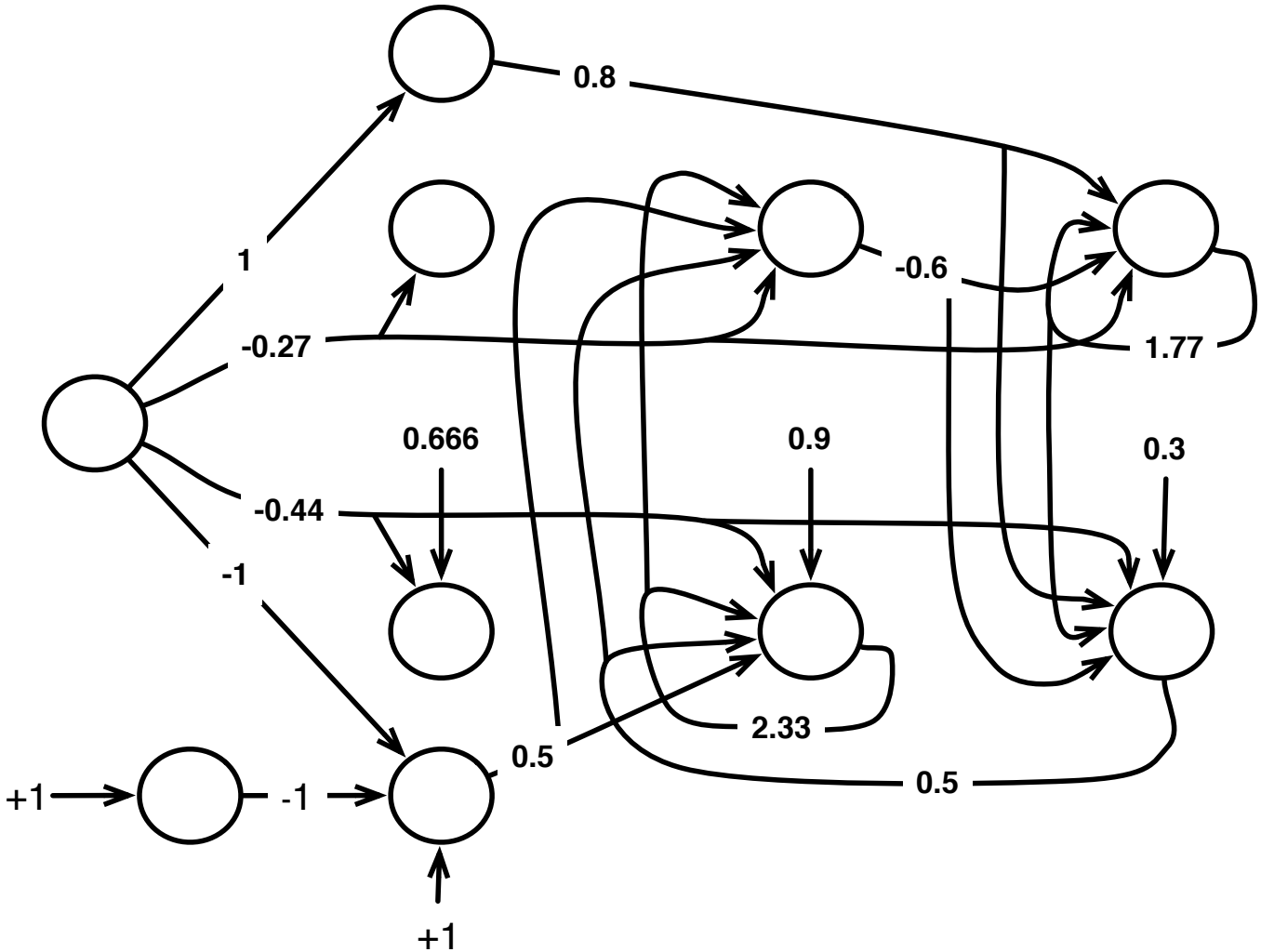
TM/poly(A)s are strictly more powerful than TMs (super-Turing) already in polynomial time of computation, i.e. $\mathbf{P/poly} \supsetneq \mathbf{P}$.

# TM vs TM/A

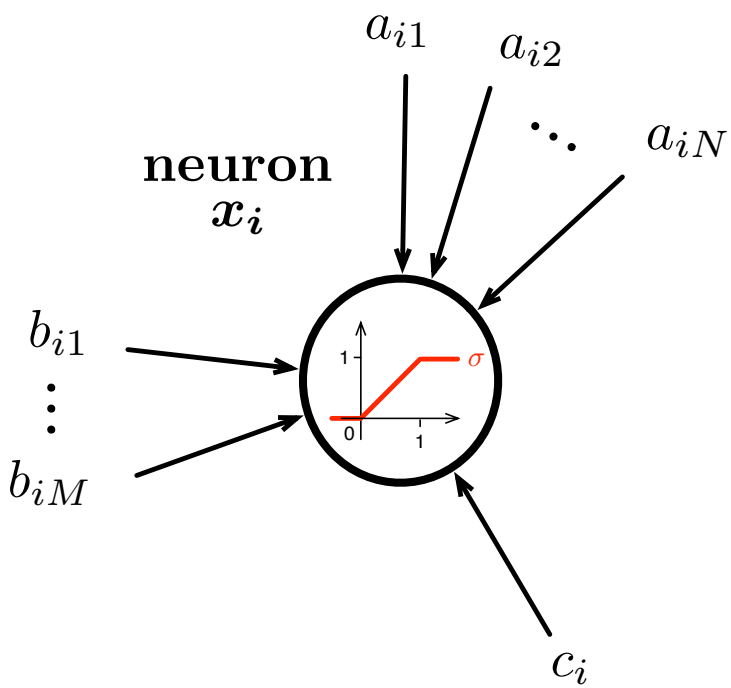
We consider Turing machines with polynomially long advice (TM/poly(A)).

TM/poly(A)s are strictly more powerful than TMs (super-Turing) already in polynomial time of computation, i.e. $\mathbf{P/poly} \supsetneq \mathbf{P}$.

# Recurrent Neural Networks



# Dynamics: static synaptic weights




$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

# Previous Results

- **Static-RNN[$\mathbb{Q}$]s**: networks with static *rational* weights
- **Static-RNN[$\mathbb{R}$]s (analog)**: networks with static *real* weights

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{Q}$]s are Turing equivalent.

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{R}$]s are super-Turing, i.e. Static-RNN[$\mathbb{R}$]s are equivalent to TM/poly(A) in poly time.

# Previous Results

- **Static-RNN[$\mathbb{Q}$]s**: networks with static *rational* weights
- **Static-RNN[$\mathbb{R}$]s (analog)**: networks with static *real* weights

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{Q}$]s are Turing equivalent.

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{R}$]s are super-Turing, i.e. Static-RNN[$\mathbb{R}$]s are equivalent to TM/poly(A) in poly time.

# Previous Results

- **Static-RNN[$\mathbb{Q}$]s**: networks with static *rational* weights
- **Static-RNN[$\mathbb{R}$]s (analog)**: networks with static *real* weights

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{Q}$]s are Turing equivalent.

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{R}$]s are super-Turing, i.e. Static-RNN[$\mathbb{R}$]s are equivalent to TM/poly(A) in poly time.

# Previous Results

- **Static-RNN[$\mathbb{Q}$]s**: networks with static *rational* weights
- **Static-RNN[$\mathbb{R}$]s (analog)**: networks with static *real* weights

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{Q}$]s are Turing equivalent.

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{R}$]s are super-Turing, i.e. Static-RNN[$\mathbb{R}$]s are equivalent to TM/poly(A) in poly time.

# Previous Results

- **Static-RNN[$\mathbb{Q}$]s**: networks with static *rational* weights
- **Static-RNN[$\mathbb{R}$]s (analog)**: networks with static *real* weights

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{Q}$]s are Turing equivalent.

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{R}$]s are super-Turing, i.e. Static-RNN[$\mathbb{R}$]s are equivalent to TM/poly(A) in poly time.

# Previous Results

- **Static-RNN[$\mathbb{Q}$]s**: networks with static *rational* weights
- **Static-RNN[$\mathbb{R}$]s (analog)**: networks with static *real* weights

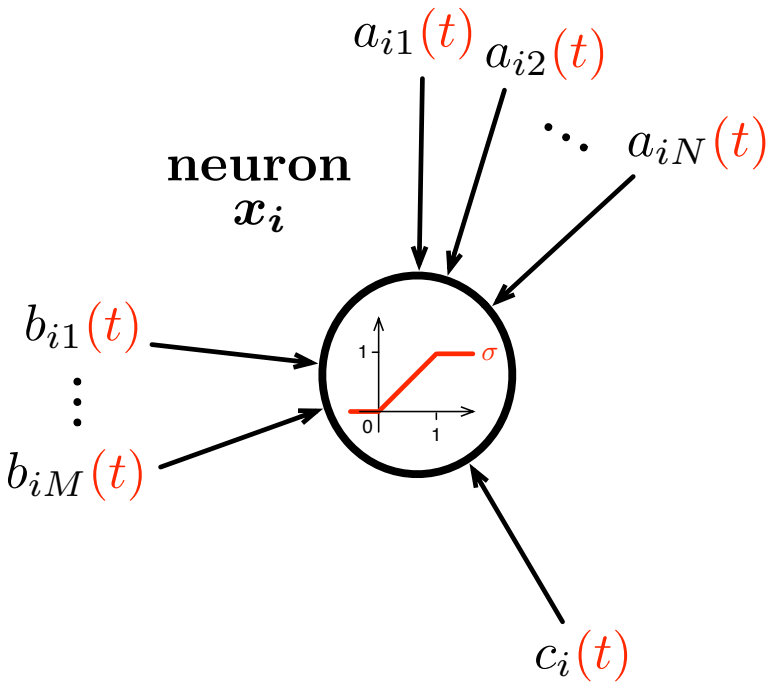
**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{Q}$]s are Turing equivalent.

**Theorem (Siegelmann & Sontag)**

Static-RNN[$\mathbb{R}$]s are super-Turing, i.e. Static-RNN[$\mathbb{R}$]s are equivalent to TM/poly(A) in poly time.

# Dynamics: evolving synaptic weights




$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t) \cdot u_j(t) + c_i(t)\right)$$

# New Results

- **Evolving-RNN[$\mathbb{Q}$]s**: networks with evolving *rational* weights
- **Evolving-RNN[$\mathbb{R}$]s (evolving analog)**: networks with evolving *real* weights

**Theorem (Cabessa & Siegelmann)**

- Evolving-RNN[$\mathbb{R}$]s are super-Turing.
- Evolving-RNN[$\mathbb{Q}$]s are super-Turing.

# New Results

- **Evolving-RNN[$\mathbb{Q}$]s**: networks with evolving *rational* weights
- **Evolving-RNN[$\mathbb{R}$]s (evolving analog)**: networks with evolving *real* weights

**Theorem (Cabessa & Siegelmann)**

- Evolving-RNN[$\mathbb{R}$]s are super-Turing.
- Evolving-RNN[$\mathbb{Q}$]s are super-Turing.

# New Results

- **Evolving-RNN[$\mathbb{Q}$]s**: networks with evolving *rational* weights
- **Evolving-RNN[$\mathbb{R}$]s (evolving analog)**: networks with evolving *real* weights

**Theorem (Cabessa & Siegelmann)**

- Evolving-RNN[$\mathbb{R}$]s are super-Turing.
- Evolving-RNN[$\mathbb{Q}$]s are super-Turing.

# New Results

- **Evolving-RNN[$\mathbb{Q}$]s**: networks with evolving *rational* weights
- **Evolving-RNN[$\mathbb{R}$]s (evolving analog)**: networks with evolving *real* weights

**Theorem (Cabessa & Siegelmann)**

- Evolving-RNN[$\mathbb{R}$]s are super-Turing.
- Evolving-RNN[$\mathbb{Q}$]s are super-Turing.

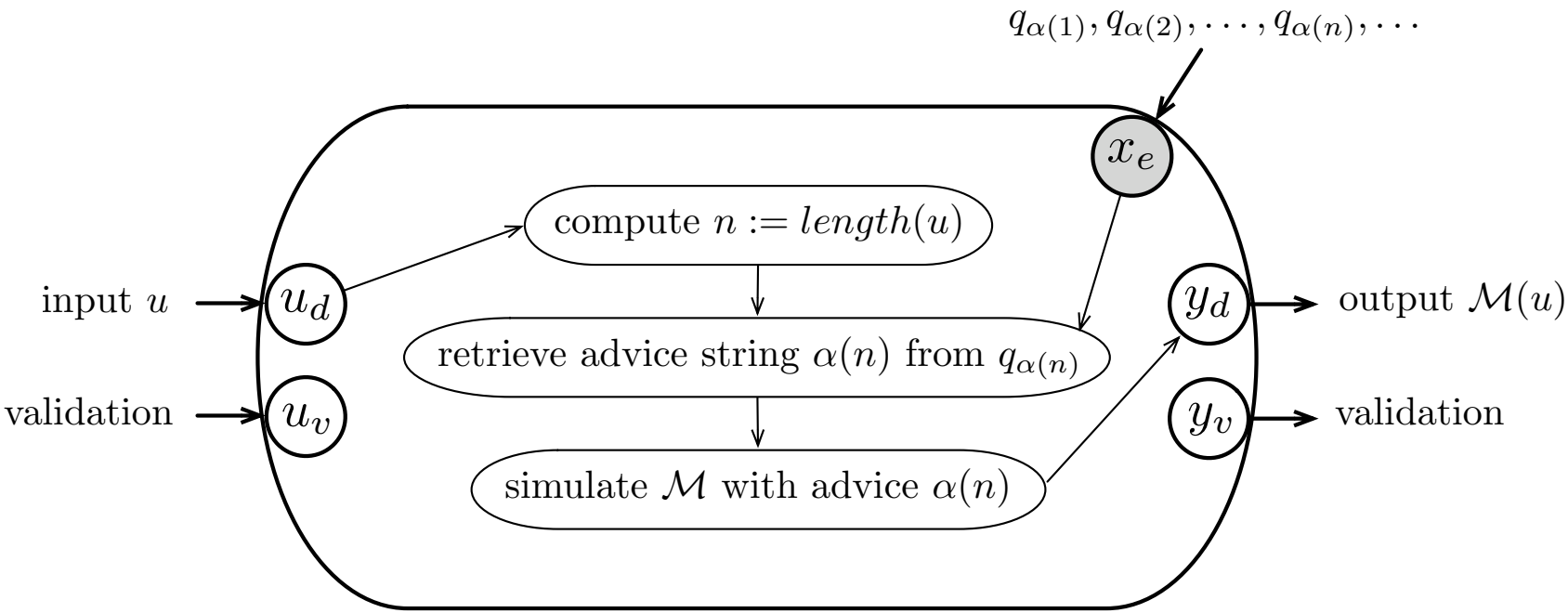
# New Results

- **Evolving-RNN[$\mathbb{Q}$]s**: networks with evolving *rational* weights
- **Evolving-RNN[$\mathbb{R}$]s (evolving analog)**: networks with evolving *real* weights

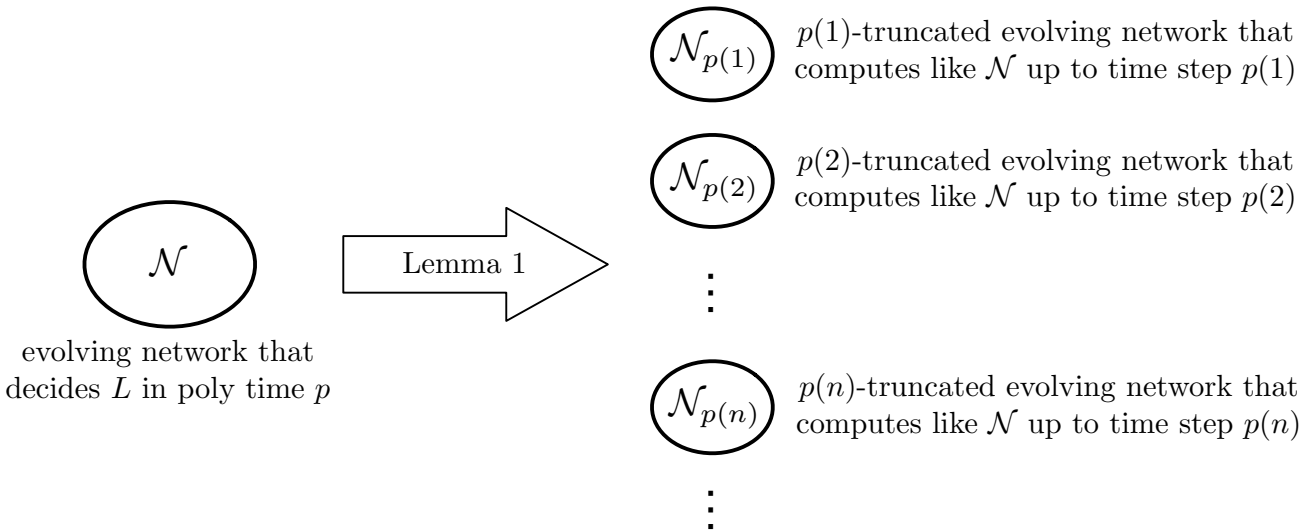
**Theorem (Cabessa & Siegelmann)**

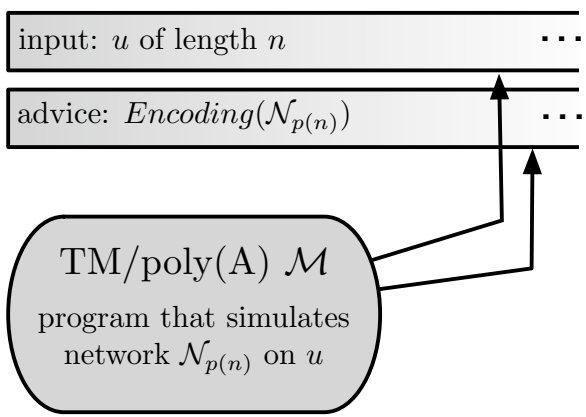
- Evolving-RNN[$\mathbb{R}$]s are super-Turing.
- Evolving-RNN[$\mathbb{Q}$]s are super-Turing.

# Proof: from TM/poly(A) to Evolving-RNN[$\mathbb{Q}$]



# Proof: from Evolving-RNN[$\mathbb{Q}$] to TM/poly(A)



# Summary

| | Static | Evolving |
|---|---|---|
| $\mathbb{Q}$ | Turing | Super-Turing |
| $\mathbb{R}$ | Super-Turing | Super-Turing |

# Conclusions

- Evolving-RNNs provide a natural abstract computational model beyond the Turing limits.
- *Architectural Evolution* is an alternating way to the *power of the continuum* to achieve super-Turing capabilities.
- The results support the idea that *architectural evolution* might play a crucial role in the computational capabilities of biological neural networks.
- Future work: study the computational power of other kinds of architecturally evolving networks: other sigmoid functions, other learning rules, spiking networks, etc.

# Conclusions

- Evolving-RNNs provide a natural abstract computational model beyond the Turing limits.
- *Architectural Evolution* is an alternating way to the *power of the continuum* to achieve super-Turing capabilities.
- The results support the idea that *architectural evolution* might play a crucial role in the computational capabilities of biological neural networks.
- Future work: study the computational power of other kinds of architecturally evolving networks: other sigmoid functions, other learning rules, spiking networks, etc.

# Conclusions

- Evolving-RNNs provide a natural abstract computational model beyond the Turing limits.
- *Architectural Evolution* is an alternating way to the *power of the continuum* to achieve super-Turing capabilities.
- The results support the idea that *architectural evolution* might play a crucial role in the computational capabilities of biological neural networks.
- Future work: study the computational power of other kinds of architecturally evolving networks: other sigmoid functions, other learning rules, spiking networks, etc.

# Conclusions

- Evolving-RNNs provide a natural abstract computational model beyond the Turing limits.
- *Architectural Evolution* is an alternating way to the *power of the continuum* to achieve super-Turing capabilities.
- The results support the idea that *architectural evolution* might play a crucial role in the computational capabilities of biological neural networks.
- Future work: study the computational power of other kinds of architecturally evolving networks: other sigmoid functions, other learning rules, spiking networks, etc.