

EXPRESSIVE POWER OF RECURRENT NEURAL NETWORKS OVER INFINITE INPUTS

J  r  mie Cabessa

Laboratoire DAVID, UVSQ Paris-Saclay

17 November 2022

INTRODUCTION

- ▶ In machine learning, artificial neural networks have proven very successful.
- ▶ In theoretical computer science, the computational power of neural networks have mainly been studied in the context of classical computation (McCulloch & Pitts, Turing, Kleene, von Neumann, Minsky, Papert,..., Siegelmann & Sontag,...).
- ▶ Following ω -automata theory, we consider neural networks involved in infinite computations.
- ▶ Related to non-terminating systems: hardwares, operating systems, control systems, etc.

INTRODUCTION

- ▶ In machine learning, artificial neural networks have proven very successful.
- ▶ In theoretical computer science, the computational power of neural networks have mainly been studied in the context of classical computation (McCulloch & Pitts, Turing, Kleene, von Neumann, Minsky, Papert,..., Siegelmann & Sontag,...).
- ▶ Following ω -automata theory, we consider neural networks involved in infinite computations.
- ▶ Related to non-terminating systems: hardwares, operating systems, control systems, etc.

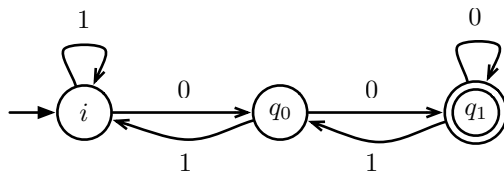
INTRODUCTION

- ▶ In machine learning, artificial neural networks have proven very successful.
- ▶ In theoretical computer science, the computational power of neural networks have mainly been studied in the context of classical computation (McCulloch & Pitts, Turing, Kleene, von Neumann, Minsky, Papert,..., Siegelmann & Sontag,...).
- ▶ Following ω -automata theory, we consider neural networks involved in infinite computations.
- ▶ Related to non-terminating systems: hardwares, operating systems, control systems, etc.

INTRODUCTION

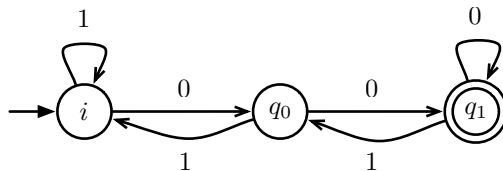
- ▶ In machine learning, artificial neural networks have proven very successful.
- ▶ In theoretical computer science, the computational power of neural networks have mainly been studied in the context of classical computation (McCulloch & Pitts, Turing, Kleene, von Neumann, Minsky, Papert,..., Siegelmann & Sontag,...).
- ▶ Following ω -automata theory, we consider neural networks involved in infinite computations.
- ▶ Related to non-terminating systems: hardwares, operating systems, control systems, etc.

FINITE STATE AUTOMATON



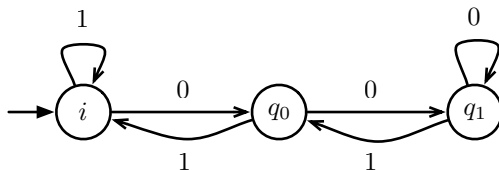
- ▶ an input u is *accepted* by \mathcal{A} if $\mathcal{A}(u)$ reaches a final state
- ▶ an input u is *rejected* by \mathcal{A} otherwise

FINITE STATE AUTOMATON



- ▶ an input u is *accepted* by \mathcal{A} if $\mathcal{A}(u)$ reaches a final state
- ▶ an input u is *rejected* by \mathcal{A} otherwise

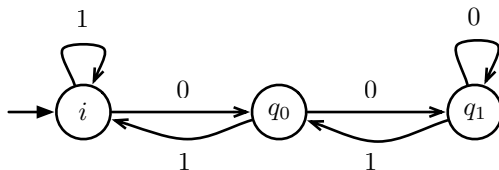
MULLER AUTOMATON



$$\mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

- ▶ an input u is *accepted* by \mathcal{A} if $\inf(\rho_u) \in \mathcal{T}$
- ▶ an input u is *rejected* by \mathcal{A} otherwise

MULLER AUTOMATON

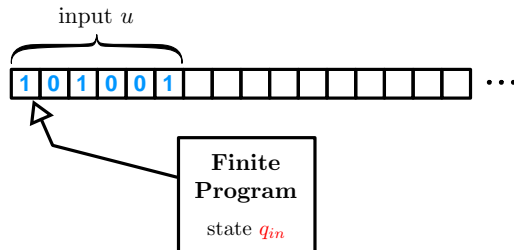


$$\mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

- ▶ an input u is *accepted* by \mathcal{A} if $\inf(\rho_u) \in \mathcal{T}$
- ▶ an input u is *rejected* by \mathcal{A} otherwise

TURING MACHINE

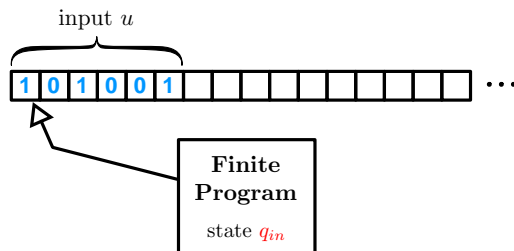
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- ▶ input u is *accepted* by \mathcal{M} if $\mathcal{M}(u)$ reaches the state q_{acc}
- ▶ input u is *rejected* by \mathcal{M} if $\mathcal{M}(u)$ reaches the state q_{rej}

TURING MACHINE

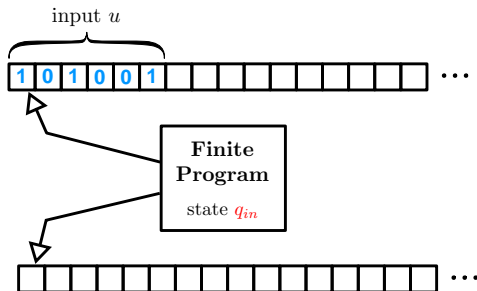
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- ▶ input u is *accepted* by \mathcal{M} if $\mathcal{M}(u)$ reaches the state q_{acc}
- ▶ input u is *rejected* by \mathcal{M} if $\mathcal{M}(u)$ reaches the state q_{rej}

TURING MACHINE WITH ADVICE

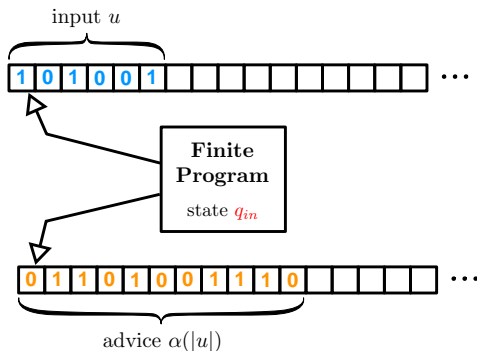
A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \rightarrow \{0, 1\}^*$.



- **P/poly** is the class of languages recognized in polynomial time by Turing machines with polynomial advices (TM/poly(A)).

TURING MACHINE WITH ADVICE

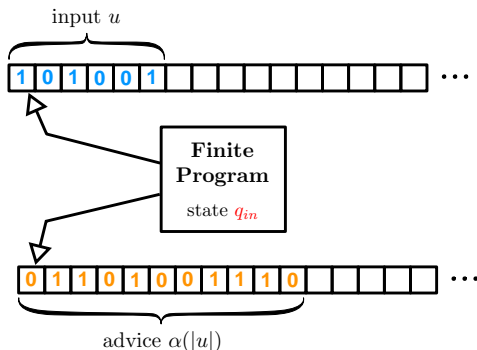
A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \rightarrow \{0, 1\}^*$.



- **P/poly** is the class of languages recognized in polynomial time by Turing machines with polynomial advices (TM/poly(A)).

TURING MACHINE WITH ADVICE

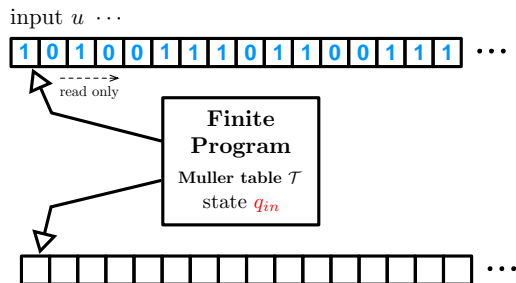
A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \rightarrow \{0, 1\}^*$.



- **P/poly** is the class of languages recognized in polynomial time by Turing machines with polynomial advices (TM/poly(A)).

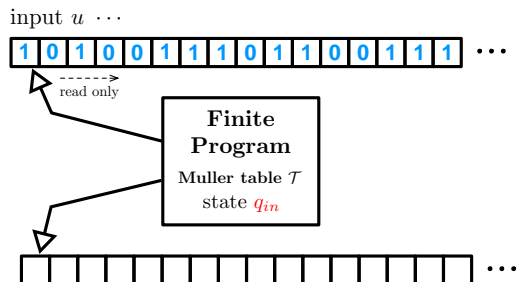
MULLER TURING MACHINE

A *Muller Turing machine* consists of a classical TM with Muller acceptance condition.



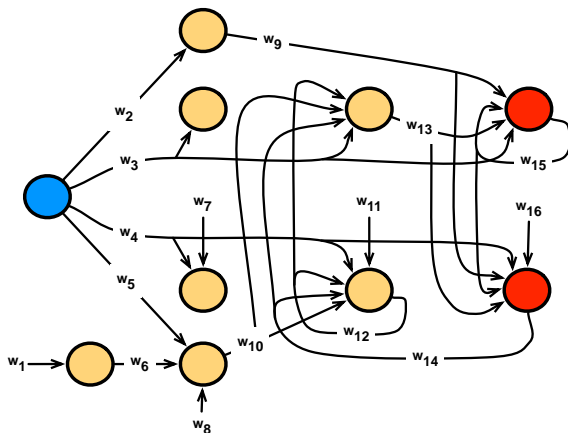
MULLER TURING MACHINE

A *Muller Turing machine* consists of a classical TM with Muller acceptance condition.

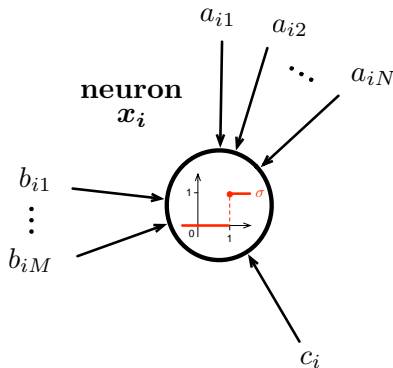


- ▶ input u is *accepted* by \mathcal{M} if $\inf(\rho_u) \in \mathcal{T}$
- ▶ input u is *rejected* by \mathcal{M} otherwise

RECURRENT NEURAL NETWORK

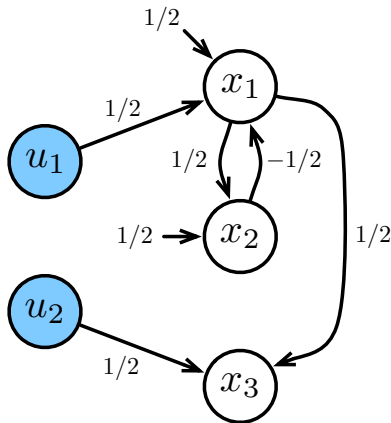


BOOLEAN RECURRENT NEURAL NETWORKS

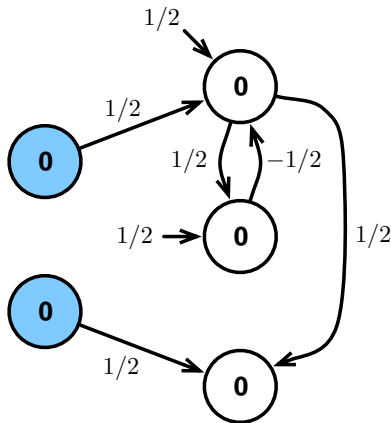


$$x_i(t+1) = \theta \left(\sum_{j=1}^N a_{ij} \cdot x_j(t) + \sum_{j=1}^M b_{ij} \cdot u_j(t) + c_i \right)$$

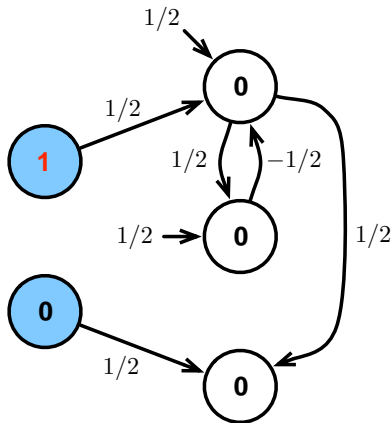
BOOLEAN RECURRENT NEURAL NETWORKS



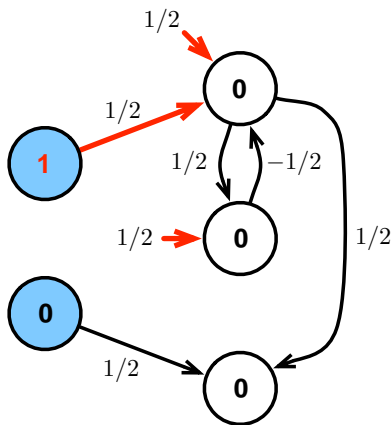
BOOLEAN RECURRENT NEURAL NETWORKS



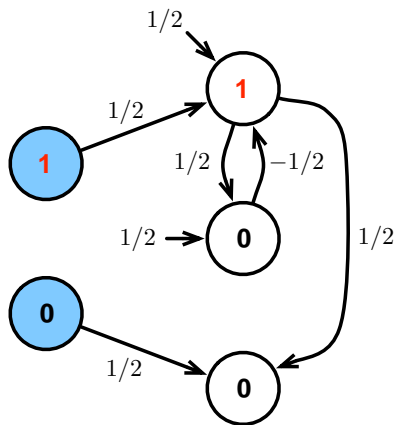
BOOLEAN RECURRENT NEURAL NETWORKS



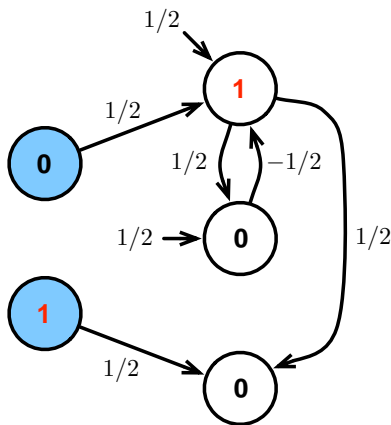
BOOLEAN RECURRENT NEURAL NETWORKS



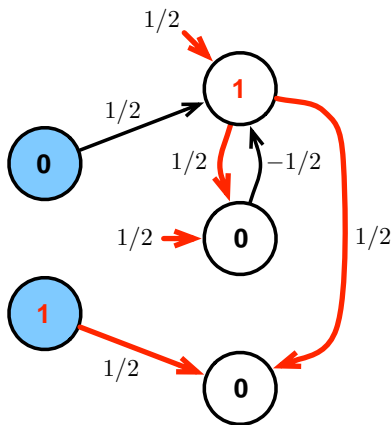
BOOLEAN RECURRENT NEURAL NETWORKS



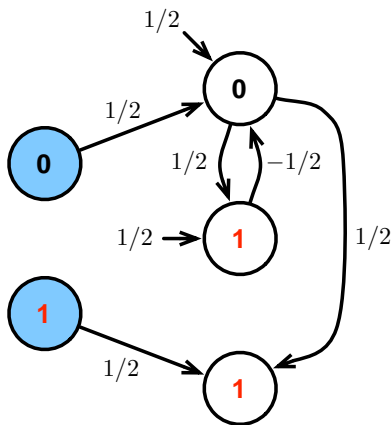
BOOLEAN RECURRENT NEURAL NETWORKS



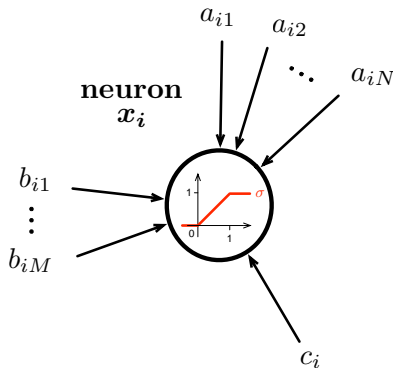
BOOLEAN RECURRENT NEURAL NETWORKS



BOOLEAN RECURRENT NEURAL NETWORKS

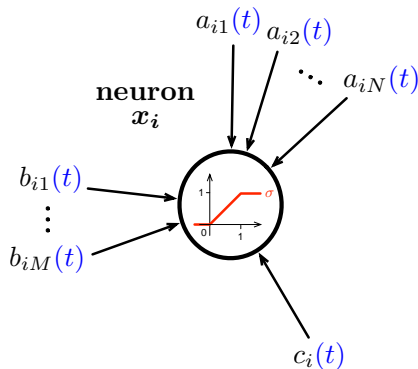


SIGMOIDAL RECURRENT NEURAL NETWORKS



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

SIGMOIDAL RECURRENT NEURAL NETWORKS



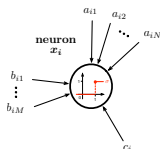
$$x_i(t+1) = \sigma \left(\sum_{j=1}^N a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^M b_{ij}(t) \cdot u_j(t) + c_i(t) \right)$$

RECURRENT NEURAL NETWORKS

1. Boolean rational RNNs: B-RNN[\mathbb{Q}]s
2. Boolean real RNNs: B-RNN[\mathbb{R}]s
3. Sigmoidal static rational RNNs: St-RNN[\mathbb{Q}]s
4. Sigmoidal static real RNNs: St-RNN[\mathbb{R}]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev₂-RNN[\mathbb{Q}]s
6. Sigmoidal bi-valued evolving real RNNs: Ev₂-RNN[\mathbb{R}]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[\mathbb{Q}]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[\mathbb{R}]s

RECURRENT NEURAL NETWORKS

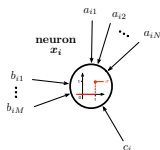
1. Boolean rational RNNs: B-RNN[\mathbb{Q}]s
2. Boolean real RNNs: B-RNN[\mathbb{R}]s
3. Sigmoidal static rational RNNs: St-RNN[\mathbb{Q}]s
4. Sigmoidal static real RNNs: St-RNN[\mathbb{R}]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev₂-RNN[\mathbb{Q}]s
6. Sigmoidal bi-valued evolving real RNNs: Ev₂-RNN[\mathbb{R}]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[\mathbb{Q}]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[\mathbb{R}]s



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

RECURRENT NEURAL NETWORKS

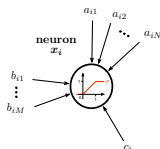
1. Boolean rational RNNs: B-RNN[\mathbb{Q}]s
2. Boolean real RNNs: B-RNN[\mathbb{R}]s
3. Sigmoidal static rational RNNs: St-RNN[\mathbb{Q}]s
4. Sigmoidal static real RNNs: St-RNN[\mathbb{R}]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev₂-RNN[\mathbb{Q}]s
6. Sigmoidal bi-valued evolving real RNNs: Ev₂-RNN[\mathbb{R}]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[\mathbb{Q}]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[\mathbb{R}]s



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

RECURRENT NEURAL NETWORKS

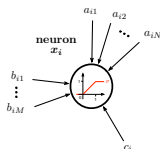
1. Boolean rational RNNs: B-RNN[\mathbb{Q}]s
2. Boolean real RNNs: B-RNN[\mathbb{R}]s
3. Sigmoidal static rational RNNs: St-RNN[\mathbb{Q}]s
4. Sigmoidal static real RNNs: St-RNN[\mathbb{R}]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev₂-RNN[\mathbb{Q}]s
6. Sigmoidal bi-valued evolving real RNNs: Ev₂-RNN[\mathbb{R}]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[\mathbb{Q}]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[\mathbb{R}]s



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

RECURRENT NEURAL NETWORKS

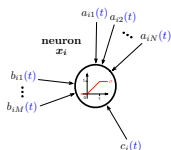
1. Boolean rational RNNs: B-RNN[\mathbb{Q}]s
2. Boolean real RNNs: B-RNN[\mathbb{R}]s
3. Sigmoidal static rational RNNs: St-RNN[\mathbb{Q}]s
4. Sigmoidal static real RNNs: St-RNN[\mathbb{R}]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev₂-RNN[\mathbb{Q}]s
6. Sigmoidal bi-valued evolving real RNNs: Ev₂-RNN[\mathbb{R}]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[\mathbb{Q}]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[\mathbb{R}]s



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

RECURRENT NEURAL NETWORKS

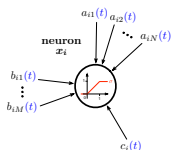
1. Boolean rational RNNs: B-RNN[\mathbb{Q}]s
2. Boolean real RNNs: B-RNN[\mathbb{R}]s
3. Sigmoidal static rational RNNs: St-RNN[\mathbb{Q}]s
4. Sigmoidal static real RNNs: St-RNN[\mathbb{R}]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev₂-RNN[\mathbb{Q}]s
6. Sigmoidal bi-valued evolving real RNNs: Ev₂-RNN[\mathbb{R}]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[\mathbb{Q}]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[\mathbb{R}]s



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

RECURRENT NEURAL NETWORKS

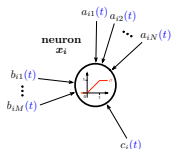
1. Boolean rational RNNs: B-RNN[\mathbb{Q}]s
2. Boolean real RNNs: B-RNN[\mathbb{R}]s
3. Sigmoidal static rational RNNs: St-RNN[\mathbb{Q}]s
4. Sigmoidal static real RNNs: St-RNN[\mathbb{R}]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev₂-RNN[\mathbb{Q}]s
6. Sigmoidal bi-valued evolving real RNNs: Ev₂-RNN[\mathbb{R}]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[\mathbb{Q}]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[\mathbb{R}]s



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

RECURRENT NEURAL NETWORKS

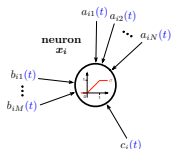
- | | |
|--|---------------------------------------|
| 1. Boolean rational RNNs: | B-RNN[\mathbb{Q}]s |
| 2. Boolean real RNNs: | B-RNN[\mathbb{R}]s |
| 3. Sigmoidal static rational RNNs: | St-RNN[\mathbb{Q}]s |
| 4. Sigmoidal static real RNNs: | St-RNN[\mathbb{R}]s |
| 5. Sigmoidal bi-valued evolving rational RNNs: | Ev ₂ -RNN[\mathbb{Q}]s |
| 6. Sigmoidal bi-valued evolving real RNNs: | Ev ₂ -RNN[\mathbb{R}]s |
| 7. Sigmoidal general evolving rational RNNs: | Ev-RNN[\mathbb{Q}]s |
| 8. Sigmoidal general evolving real N-RNNs: | Ev-RNN[\mathbb{R}]s |



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

RECURRENT NEURAL NETWORKS

- | | |
|--|---------------------------------------|
| 1. Boolean rational RNNs: | B-RNN[\mathbb{Q}]s |
| 2. Boolean real RNNs: | B-RNN[\mathbb{R}]s |
| 3. Sigmoidal static rational RNNs: | St-RNN[\mathbb{Q}]s |
| 4. Sigmoidal static real RNNs: | St-RNN[\mathbb{R}]s |
| 5. Sigmoidal bi-valued evolving rational RNNs: | Ev ₂ -RNN[\mathbb{Q}]s |
| 6. Sigmoidal bi-valued evolving real RNNs: | Ev ₂ -RNN[\mathbb{R}]s |
| 7. Sigmoidal general evolving rational RNNs: | Ev-RNN[\mathbb{Q}]s |
| 8. Sigmoidal general evolving real N-RNNs: | Ev-RNN[\mathbb{R}]s |



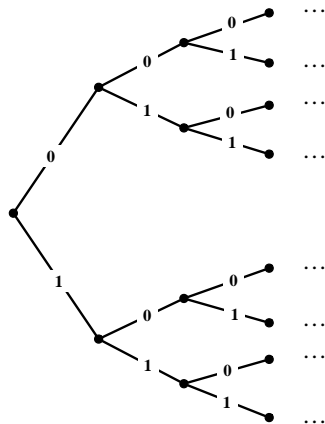
$$x_i(t+1) = \sigma \left(\sum_{j=1}^N a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^M b_{ij}(t) \cdot u_j(t) + c_i(t) \right)$$

RESULTS (CLASSICAL COMPUTATION)

	BOOLEAN	STATIC	BI-VALUED	EVOLVING	EVOLVING
\mathbb{Q}	FSA	TM	TM/poly(A)	TM/poly(A)	
	REG	P	P/poly	P/poly	
	KI 56, Mi 67	S&S 95	C&S 11,14	C&S 11,14	
\mathbb{R}	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)	
	REG	P/poly	P/poly	P/poly	
	KI 56, Mi 67	S&S 94	C&S 11,14	C&S 11,14	

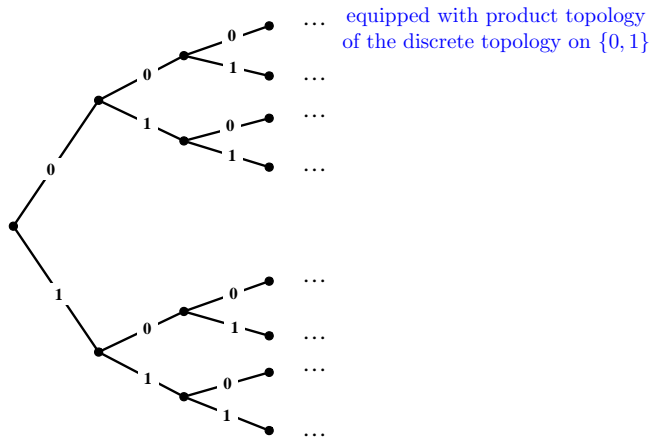
CANTOR SPACE

The Cantor space $\{0, 1\}^\omega$
the set of infinite sequences of bits



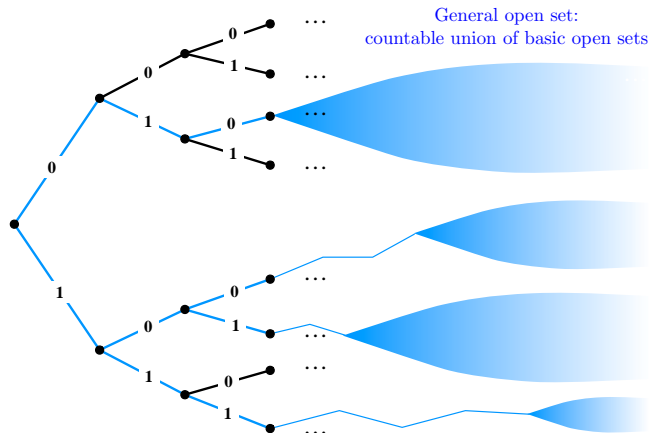
CANTOR SPACE

The Cantor space $\{0, 1\}^\omega$
the set of infinite sequences of bits



CANTOR SPACE

The Cantor space $\{0, 1\}^\omega$
the set of infinite sequences of bits



BOREL HIERARCHY

height ω_1

•
•
•

BOREL HIERARCHY

height ω_1

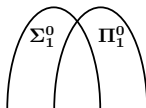
•
•
•

 Σ_1^0

BOREL HIERARCHY

height ω_1

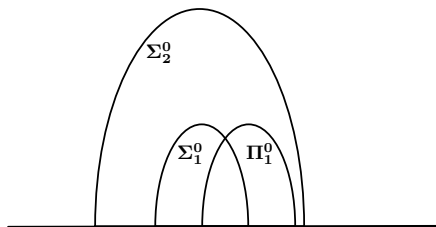
•
•
•



BOREL HIERARCHY

height ω_1

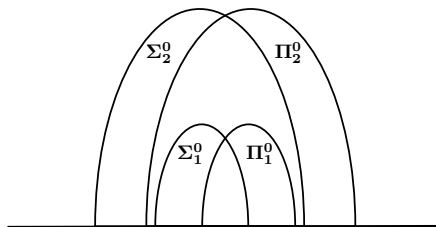
⋮



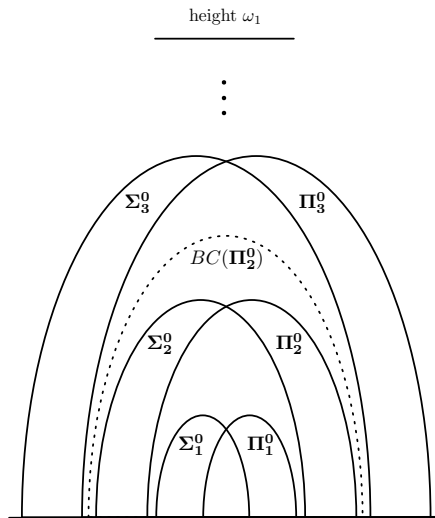
BOREL HIERARCHY

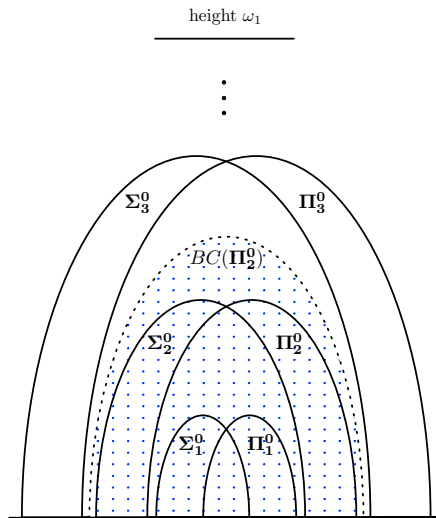
height ω_1

⋮

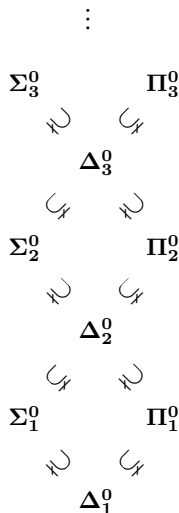


BOREL HIERARCHY



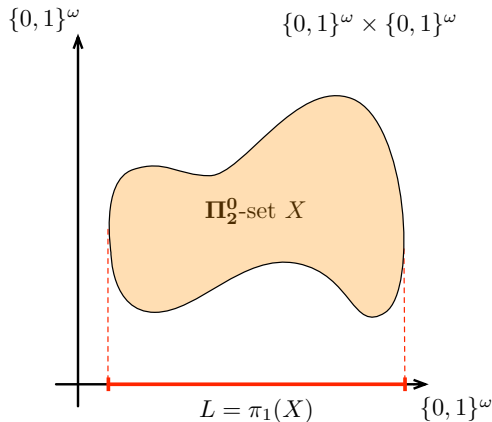


BOREL HIERARCHY



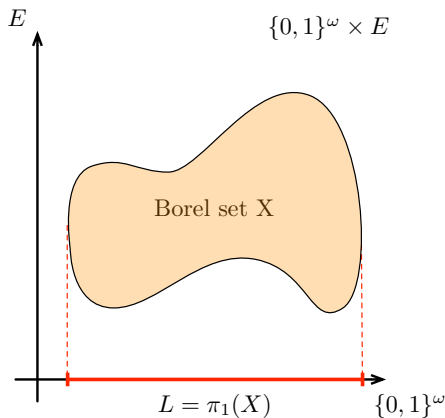
ANALYTIC SET

- An ω -language $L \subseteq \{0,1\}^\omega$ is *analytic* (Σ_1^1) iff it is the first projection of some Π_2^0 -set $X \subseteq \{0,1\}^\omega \times \{0,1\}^\omega$.

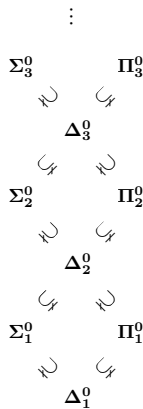


ANALYTIC SET

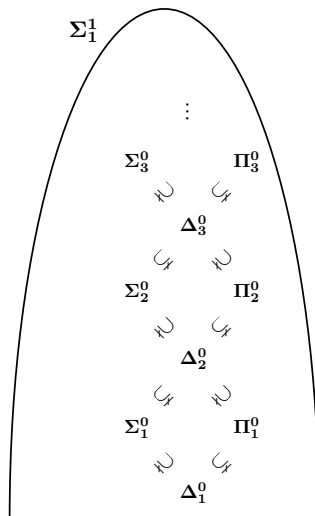
- An ω -language $L \subseteq \{0,1\}^\omega$ is *analytic* (Σ_1^1) iff it is the first projection of some Borel set $X \subseteq \{0,1\}^\omega \times E$, where E is a Polish space (separable completely metrizable topological space).

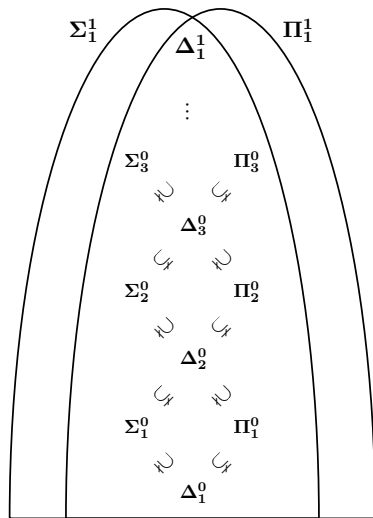


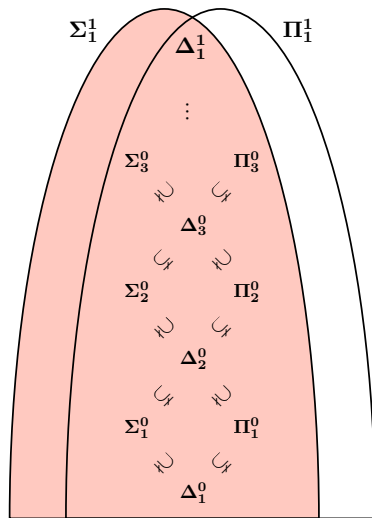
BOREL HIERARCHY AND ANALYTIC CLASS

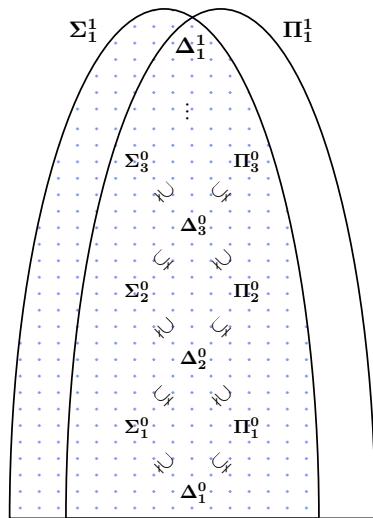


BOREL HIERARCHY AND ANALYTIC CLASS



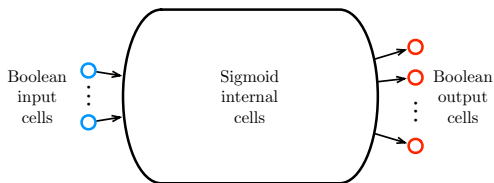






DETERMINISTIC ω -RNNs

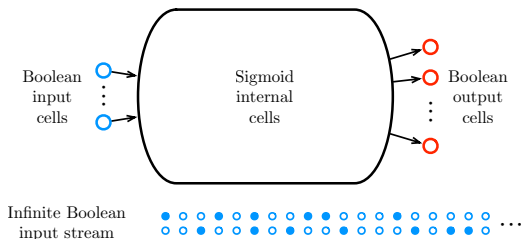
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite inputs.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ accepted by \mathcal{N} iff $\mathcal{N}(s)$ enters a meaningful attractor.

DETERMINISTIC ω -RNNs

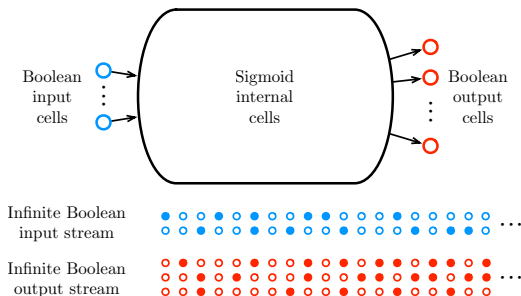
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite inputs.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ accepted by \mathcal{N} iff $\mathcal{N}(s)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ rejected by \mathcal{N} otherwise.

DETERMINISTIC ω -RNNs

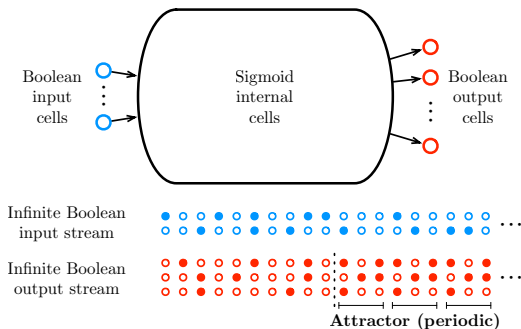
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite inputs.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ accepted by \mathcal{N} iff $\mathcal{N}(s)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ rejected by \mathcal{N} otherwise.

DETERMINISTIC ω -RNNs

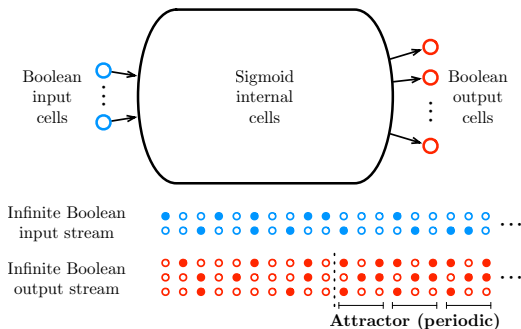
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite inputs.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ accepted by \mathcal{N} iff $\mathcal{N}(s)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ rejected by \mathcal{N} otherwise.

DETERMINISTIC ω -RNNs

We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite inputs.



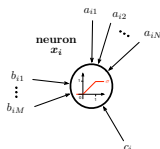
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by \mathcal{N} iff $\mathcal{N}(s)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *rejected* by \mathcal{N} otherwise.

DETERMINISTIC ω -RNNs

1. static rational RNNs: D-St-RNN[\mathbb{Q}]s
2. static real RNNs: D-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: D-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: D-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: D-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: D-Ev-RNN[\mathbb{R}]s

DETERMINISTIC ω -RNNs

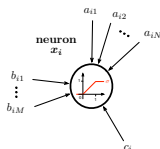
1. static rational RNNs: D-St-RNN[\mathbb{Q}]s
2. static real RNNs: D-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: D-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: D-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: D-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: D-Ev-RNN[\mathbb{R}]s



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

DETERMINISTIC ω -RNNs

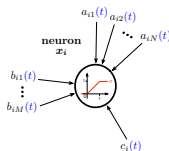
1. static rational RNNs: D-St-RNN[\mathbb{Q}]s
2. static real RNNs: D-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: D-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: D-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: D-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: D-Ev-RNN[\mathbb{R}]s



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

DETERMINISTIC ω -RNNs

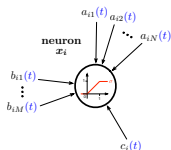
1. static rational RNNs: D-St-RNN[\mathbb{Q}]s
2. static real RNNs: D-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: D-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: D-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: D-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: D-Ev-RNN[\mathbb{R}]s



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

DETERMINISTIC ω -RNNs

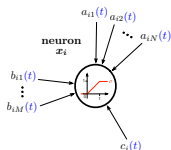
1. static rational RNNs: D-St-RNN[\mathbb{Q}]s
2. static real RNNs: D-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: D-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: D-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: D-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: D-Ev-RNN[\mathbb{R}]s



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

DETERMINISTIC ω -RNNs

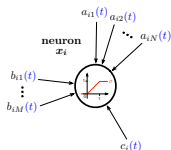
1. static rational RNNs: D-St-RNN[\mathbb{Q}]s
2. static real RNNs: D-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: D-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: D-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: D-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: D-Ev-RNN[\mathbb{R}]s



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

DETERMINISTIC ω -RNNs

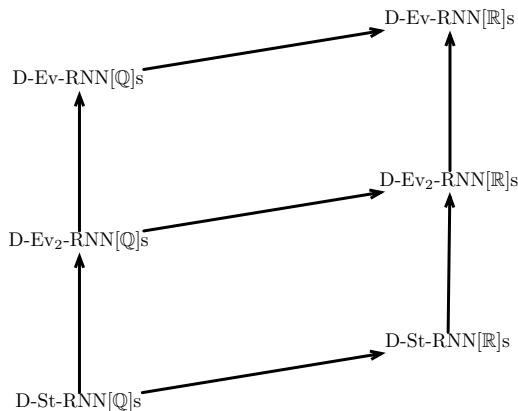
1. static rational RNNs: D-St-RNN[\mathbb{Q}]s
2. static real RNNs: D-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: D-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: D-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: D-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: D-Ev-RNN[\mathbb{R}]s



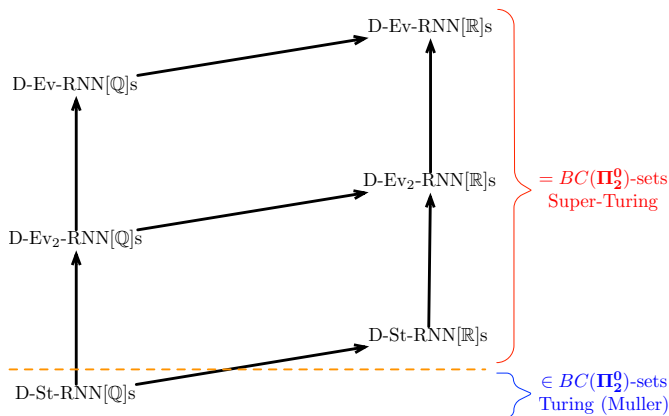
$$x_i(t+1) = \sigma \left(\sum_{j=1}^N a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^M b_{ij}(t) \cdot u_j(t) + c_i(t) \right)$$

DETERMINISTIC ω -RNNs

Relationships between those models:



RESULTS



RESULTS

THEOREM

Let $L \subseteq (\mathbb{B}^M)^\omega$. The following conditions are equivalent.

- ▶ *L is recognizable by some deterministic Muller TM (and thus $L \in BC(\Pi_2^0)$)*
- ▶ *L is recognizable by some D-St-RNN[\mathbb{Q}]*

PROOF (SKETCH): Generalization of the classical equivalence between TMs and St-RNN[\mathbb{Q}] (Siegelmann & Sontag 95).

RESULTS

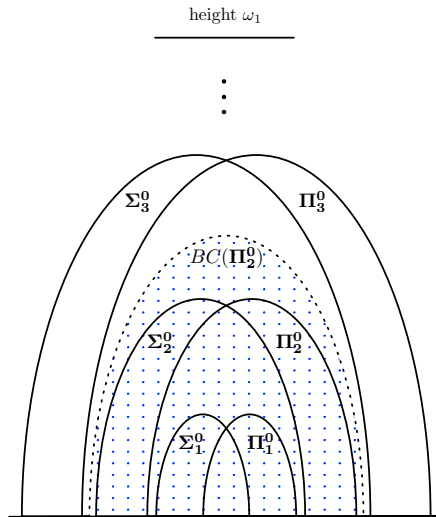
THEOREM

Let $L \subseteq (\mathbb{B}^M)^\omega$. The following conditions are equivalent.

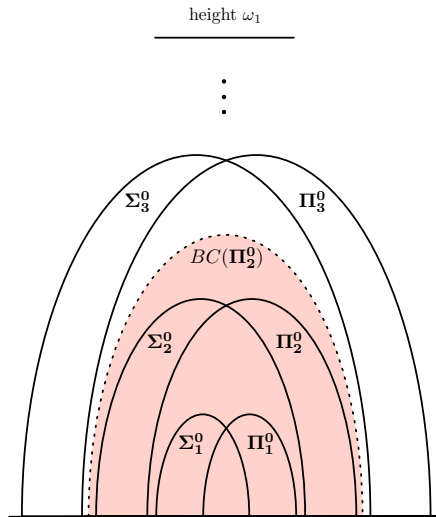
- ▶ *L is recognizable by some deterministic Muller TM (and thus $L \in BC(\Pi_2^0)$)*
- ▶ *L is recognizable by some D-St-RNN[\mathbb{Q}]*

PROOF (SKETCH): Generalization of the classical equivalence between TMs and St-RNN[\mathbb{Q}] (Siegelmann & Sontag 95).

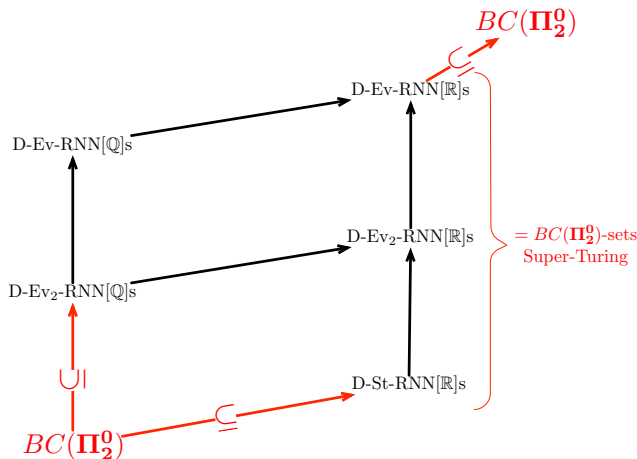
RESULTS



RESULTS



RESULTS



RESULTS

PROPOSITION

Let $L \in BC(\Pi_2^0)$. Then L is recognizable by some $D\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $D\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

- Assume that $L \in \Pi_2^0$.
- The proof can be extended to $L \in BC(\Pi_2^0)$.

RESULTS

PROPOSITION

Let $L \in BC(\Pi_2^0)$. Then L is recognizable by some $D\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $D\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

- ▶ Assume that $L \in \Pi_2^0$.
- ▶ The proof can be extended to $L \in BC(\Pi_2^0)$.
- ▶ L can be written as

$$L = \bigcap_{i \geq 0} \bigcup_{j \geq 0} p_{i,j} \cdot (\mathbb{B}^M)^\omega$$

- ▶ Hence L can be recursively encoded into some $r_L \in \mathbb{R}$.

RESULTS

PROPOSITION

Let $L \in BC(\Pi_2^0)$. Then L is recognizable by some $D\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $D\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

- ▶ Assume that $L \in \Pi_2^0$.
- ▶ The proof can be extended to $L \in BC(\Pi_2^0)$.
- ▶ L can be written as

$$L = \bigcap_{i \geq 0} \bigcup_{j \geq 0} p_{i,j} \cdot (\mathbb{B}^M)^\omega$$

- ▶ Hence L can be recursively encoded into some $r_L \in \mathbb{R}$.

RESULTS

PROPOSITION

Let $L \in BC(\Pi_2^0)$. Then L is recognizable by some $D\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $D\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

- ▶ Assume that $L \in \Pi_2^0$.
- ▶ The proof can be extended to $L \in BC(\Pi_2^0)$.
- ▶ L can be written as

$$L = \bigcap_{i \geq 0} \bigcup_{j \geq 0} p_{i,j} \cdot (\mathbb{B}^M)^\omega$$

- ▶ Hence L can be recursively encoded into some $r_L \in \mathbb{R}$.

RESULTS

PROPOSITION

Let $L \in BC(\Pi_2^0)$. Then L is recognizable by some $D\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $D\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

- ▶ Assume that $L \in \Pi_2^0$.
- ▶ The proof can be extended to $L \in BC(\Pi_2^0)$.
- ▶ L can be written as

$$L = \bigcap_{i \geq 0} \bigcup_{j \geq 0} p_{i,j} \cdot (\mathbb{B}^M)^\omega$$

- ▶ Hence L can be recursively encoded into some $r_L \in \mathbb{R}$.

RESULTS

Algorithm 1 Infinite procedure

Require: Input stream $s = \mathbf{u}(0)\mathbf{u}(1)\mathbf{u}(2) \cdots \in (\mathbb{B}^M)^\omega$ and real number r_L .

```

1: store each  $\mathbf{u}(t) \in \mathbb{B}^M$  as they arrive
2:  $i \leftarrow 0, j \leftarrow 0$ 
3: loop
4:   decode  $p_{i,j}$  from  $r_L$                                      // recursive procedure if  $r_L$  is given
5:   if  $p_{i,j} \subseteq s[0:c]$  then                                     //  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
6:     return 1                                                    //  $\exists j$  s.t.  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
7:      $i \leftarrow i + 1, j \leftarrow 0$                             // test if  $s \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega$ 
8:   else                                                         //  $s \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
9:     return 0                                                    //  $\neg \exists j' \leq j$  s.t.  $s \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega$ 
10:     $i \leftarrow i, j \leftarrow j + 1$                           // test if  $s \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega$ 
11:  end if
12: end loop

```

Algo returns ∞ -many 1's iff $s \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega) = L$.

RESULTS

Algorithm 1 Infinite procedure

Require: Input stream $s = \mathbf{u}(0)\mathbf{u}(1)\mathbf{u}(2) \cdots \in (\mathbb{B}^M)^\omega$ and real number r_L .

```

1: store each  $\mathbf{u}(t) \in \mathbb{B}^M$  as they arrive
2:  $i \leftarrow 0, j \leftarrow 0$ 
3: loop
4:   decode  $p_{i,j}$  from  $r_L$                                 // recursive procedure if  $r_L$  is given
5:   if  $p_{i,j} \subseteq s[0:c]$  then                                //  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
6:     return 1                                                //  $\exists j$  s.t.  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
7:      $i \leftarrow i + 1, j \leftarrow 0$                         // test if  $s \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega$ 
8:   else                                                        //  $s \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
9:     return 0                                                //  $\neg \exists j' \leq j$  s.t.  $s \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega$ 
10:     $i \leftarrow i, j \leftarrow j + 1$                         // test if  $s \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega$ 
11:  end if
12: end loop

```

Algo returns ∞ -many 1's iff $s \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega) = L$.

RESULTS

Algorithm 1 Infinite procedure

Require: Input stream $s = \mathbf{u}(0)\mathbf{u}(1)\mathbf{u}(2) \cdots \in (\mathbb{B}^M)^\omega$ and real number r_L .

```

1: store each  $\mathbf{u}(t) \in \mathbb{B}^M$  as they arrive
2:  $i \leftarrow 0, j \leftarrow 0$ 
3: loop
4:   decode  $p_{i,j}$  from  $r_L$                                      // recursive procedure if  $r_L$  is given
5:   if  $p_{i,j} \subseteq s[0:c]$  then                                     //  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
6:     return 1                                                    //  $\exists j$  s.t.  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
7:      $i \leftarrow i + 1, j \leftarrow 0$                              // test if  $s \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega$ 
8:   else                                                         //  $s \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
9:     return 0                                                    //  $\neg \exists j' \leq j$  s.t.  $s \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega$ 
10:     $i \leftarrow i, j \leftarrow j + 1$                              // test if  $s \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega$ 
11:  end if
12: end loop

```

Algo returns ∞ -many 1's iff $s \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega) = L$.

RESULTS

Algorithm 1 Infinite procedure

Require: Input stream $s = \mathbf{u}(0)\mathbf{u}(1)\mathbf{u}(2) \cdots \in (\mathbb{B}^M)^\omega$ and real number r_L .

```

1: store each  $\mathbf{u}(t) \in \mathbb{B}^M$  as they arrive
2:  $i \leftarrow 0, j \leftarrow 0$ 
3: loop
4:   decode  $p_{i,j}$  from  $r_L$                                      // recursive procedure if  $r_L$  is given
5:   if  $p_{i,j} \subseteq s[0:c]$  then                                     //  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
6:     return 1                                                    //  $\exists j$  s.t.  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
7:      $i \leftarrow i + 1, j \leftarrow 0$                             // test if  $s \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega$ 
8:   else                                                         //  $s \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
9:     return 0                                                    //  $\neg \exists j' \leq j$  s.t.  $s \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega$ 
10:     $i \leftarrow i, j \leftarrow j + 1$                             // test if  $s \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega$ 
11:   end if
12: end loop

```

Algo returns ∞ -many 1's iff $s \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega) = L$.

RESULTS

Algorithm 1 Infinite procedure

Require: Input stream $s = u(0)u(1)u(2) \cdots \in (\mathbb{B}^M)^\omega$ and real number r_L .

```

1: store each  $u(t) \in \mathbb{B}^M$  as they arrive
2:  $i \leftarrow 0, j \leftarrow 0$ 
3: loop
4:   decode  $p_{i,j}$  from  $r_L$                                      // recursive procedure if  $r_L$  is given
5:   if  $p_{i,j} \subseteq s[0:c]$  then                                     //  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
6:     return 1                                                    //  $\exists j$  s.t.  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
7:      $i \leftarrow i + 1, j \leftarrow 0$                             // test if  $s \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega$ 
8:   else                                                         //  $s \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
9:     return 0                                                    //  $\neg \exists j' \leq j$  s.t.  $s \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega$ 
10:     $i \leftarrow i, j \leftarrow j + 1$                             // test if  $s \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega$ 
11:  end if
12: end loop

```

Algo returns ∞ -many 1's iff $s \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega) = L$.

Algorithm 1 Infinite procedure

```

1: store each  $u(t) \in \mathbb{B}^M$  as they arrive
2:  $i \leftarrow 0, j \leftarrow 0$ 
3: loop
4:   decode  $p_{i,j}$  from  $r_L$                                      // recursive procedure if  $r_L$  is given
5:   if  $p_{i,j} \subseteq s[0:c]$  then                                   //  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
6:     return 1                                                    //  $\exists j$  s.t.  $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
7:      $i \leftarrow i + 1, j \leftarrow 0$                           // test if  $s \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega$ 
8:   else                                                         //  $s \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega$ 
9:     return 0                                                    //  $\neg \exists j' \leq j$  s.t.  $s \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega$ 
10:     $i \leftarrow i, j \leftarrow j + 1$                           // test if  $s \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega$ 
11:  end if
12: end loop

```

◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

RESULTS

- ▶ The algo can be simulated by some D-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on s iff $\mathcal{N}(s)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\mathcal{N}(s)$ visits a meaningful attractor
 - ▶ iff the algo returns infinitely many 1's on s
 - ▶ iff $s \in L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some D-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on s iff $\mathcal{N}(s)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\mathcal{N}(s)$ visits a meaningful attractor
 - ▶ iff the algo returns infinitely many 1's on s
 - ▶ iff $s \in L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some D-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on s iff $\mathcal{N}(s)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\mathcal{N}(s)$ visits a meaningful attractor
 - ▶ iff the algo returns infinitely many 1's on s
 - ▶ iff $s \in L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some D-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on s iff $\mathcal{N}(s)$ visits a meaningful
 attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\mathcal{N}(s)$ visits a meaningful attractor
 - ▶ iff the algo returns infinitely many 1's on s
 - ▶ iff $s \in L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some D-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on s iff $\mathcal{N}(s)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\mathcal{N}(s)$ visits a meaningful attractor
 - ▶ iff the algo returns infinitely many 1's on s
 - ▶ iff $s \in L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some D-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on s iff $\mathcal{N}(s)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\mathcal{N}(s)$ visits a meaningful attractor
 - ▶ iff the algo returns infinitely many 1's on s
 - ▶ iff $s \in L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

PROPOSITION

Let \mathcal{N} be some D-Ev-RNN $[\mathbb{R}]$. Then $L(\mathcal{N}) \in BC(\Pi_2^0)$.

PROOF (SKETCH):

- The function $f_{\mathcal{N}} : (\mathbb{B}^M)^\omega \rightarrow (\mathbb{B}^P)^\omega$ associated with the dynamics of \mathcal{N} is continuous (Lecture 4, Σ_1^0 & Σ_2^0).
- Accordingly, the ω -language $L(\mathcal{N})$ can be expressed as a finite intersection of preimages of Σ_2^0 sets (which are Σ_2^0 by continuity of $f_{\mathcal{N}}$).
- Therefore, $L(\mathcal{N}) \in BC(\Pi_2^0)$.

RESULTS

PROPOSITION

Let \mathcal{N} be some D -Ev-RNN $[\mathbb{R}]$. Then $L(\mathcal{N}) \in BC(\Pi_2^0)$.

PROOF (SKETCH):

- ▶ The function $f_{\mathcal{N}} : (\mathbb{B}^M)^\omega \rightarrow (\mathbb{B}^P)^\omega$ associated with the dynamics of \mathcal{N} is continuous (preimage of a Σ_1^0 is Σ_1^0).
- ▶ Accordingly, the ω -language $L(\mathcal{N})$ can be expressed as a finite intersection of preimages of Σ_2^0 sets (which are Σ_2^0 by continuity of $f_{\mathcal{N}}$).
- ▶ Therefore, $L(\mathcal{N}) \in BC(\Pi_2^0)$. □

RESULTS

PROPOSITION

Let \mathcal{N} be some D -Ev-RNN $[\mathbb{R}]$. Then $L(\mathcal{N}) \in BC(\Pi_2^0)$.

PROOF (SKETCH):

- ▶ The function $f_{\mathcal{N}} : (\mathbb{B}^M)^\omega \rightarrow (\mathbb{B}^P)^\omega$ associated with the dynamics of \mathcal{N} is continuous (preimage of a Σ_1^0 is Σ_1^0).
- ▶ Accordingly, the ω -language $L(\mathcal{N})$ can be expressed as a finite intersection of preimages of Σ_2^0 sets (which are Σ_2^0 by continuity of $f_{\mathcal{N}}$).
- ▶ Therefore, $L(\mathcal{N}) \in BC(\Pi_2^0)$. □

RESULTS

PROPOSITION

Let \mathcal{N} be some D -Ev-RNN $[\mathbb{R}]$. Then $L(\mathcal{N}) \in BC(\Pi_2^0)$.

PROOF (SKETCH):

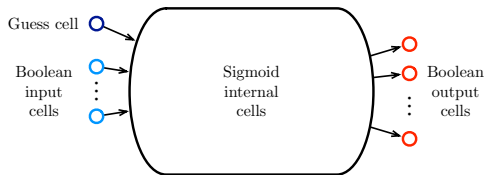
- ▶ The function $f_{\mathcal{N}} : (\mathbb{B}^M)^\omega \rightarrow (\mathbb{B}^P)^\omega$ associated with the dynamics of \mathcal{N} is continuous (preimage of a Σ_1^0 is Σ_1^0).
- ▶ Accordingly, the ω -language $L(\mathcal{N})$ can be expressed as a finite intersection of preimages of Σ_2^0 sets (which are Σ_2^0 by continuity of $f_{\mathcal{N}}$).
- ▶ Therefore, $L(\mathcal{N}) \in BC(\Pi_2^0)$. □

RESULTS – SUMMARY

DET.	STATIC	BI-VALUED EVOLVING	GENERAL EVOLVING
\mathbb{Q}	D-St-RNN[\mathbb{Q}]s $\in BC(\Pi_2^0)$ Turing (Muller)	D-Ev ₂ -RNN[\mathbb{Q}]s $= BC(\Pi_2^0)$ super-Turing	D-Ev-RNN[\mathbb{Q}]s $= BC(\Pi_2^0)$ super-Turing
\mathbb{R}	D-St-RNN[\mathbb{R}]s $= BC(\Pi_2^0)$ super-Turing	D-Ev ₂ -RNN[\mathbb{R}]s $= BC(\Pi_2^0)$ super-Turing	D-Ev-RNN[\mathbb{R}]s $= BC(\Pi_2^0)$ super-Turing

NONDETERMINISM OF TYPE I

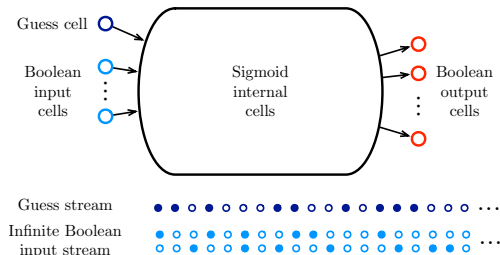
The RNNs are provided with an additional Boolean guess cell.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by \mathcal{N} iff there exists some guess $g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ enters a meaningful attractor.

NONDETERMINISM OF TYPE I

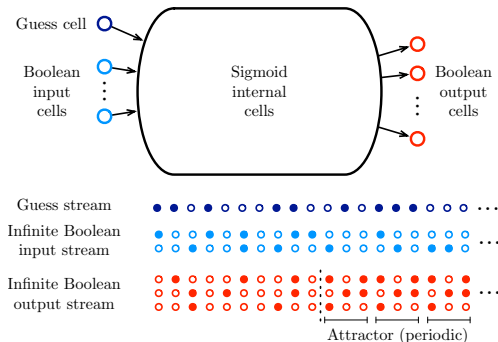
The RNNs are provided with an additional Boolean guess cell.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by \mathcal{N} iff there exists some guess $g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *rejected* by \mathcal{N} otherwise.

NONDETERMINISM OF TYPE I

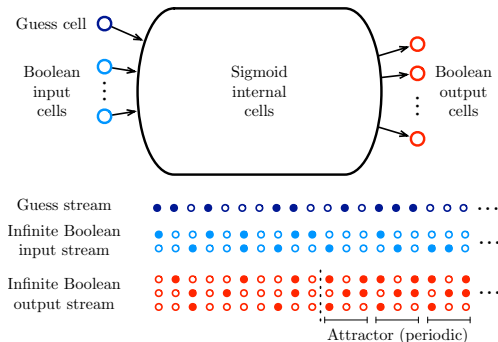
The RNNs are provided with an additional Boolean guess cell.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by \mathcal{N} iff there exists some guess $g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *rejected* by \mathcal{N} otherwise.

NONDETERMINISM OF TYPE I

The RNNs are provided with an additional Boolean guess cell.



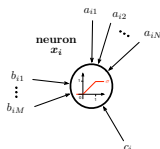
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by \mathcal{N} iff there exists some guess $g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *rejected* by \mathcal{N} otherwise.

NONDETERMINISTIC ω -RNNs: TYPE I

1. static rational RNNs: N-St-RNN[\mathbb{Q}]s
2. static real RNNs: N-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: N-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: N-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: N-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: N-Ev-RNN[\mathbb{R}]s

NONDETERMINISTIC ω -RNNs: TYPE I

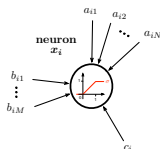
1. static rational RNNs: N-St-RNN[\mathbb{Q}]s
2. static real RNNs: N-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: N-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: N-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: N-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: N-Ev-RNN[\mathbb{R}]s



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

NONDETERMINISTIC ω -RNNs: TYPE I

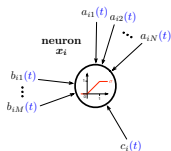
1. static rational RNNs: N-St-RNN[\mathbb{Q}]s
2. static real RNNs: N-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: N-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: N-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: N-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: N-Ev-RNN[\mathbb{R}]s



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

NONDETERMINISTIC ω -RNNs: TYPE I

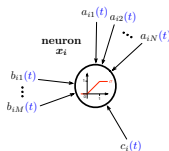
1. static rational RNNs: N-St-RNN[\mathbb{Q}]s
2. static real RNNs: N-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: N-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: N-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: N-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: N-Ev-RNN[\mathbb{R}]s



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

NONDETERMINISTIC ω -RNNs: TYPE I

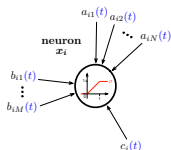
1. static rational RNNs: N-St-RNN[\mathbb{Q}]s
2. static real RNNs: N-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: N-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: N-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: N-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: N-Ev-RNN[\mathbb{R}]s



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

NONDETERMINISTIC ω -RNNs: TYPE I

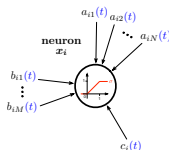
1. static rational RNNs: N-St-RNN[\mathbb{Q}]s
2. static real RNNs: N-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: N-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: N-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: N-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: N-Ev-RNN[\mathbb{R}]s



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

NONDETERMINISTIC ω -RNNs: TYPE I

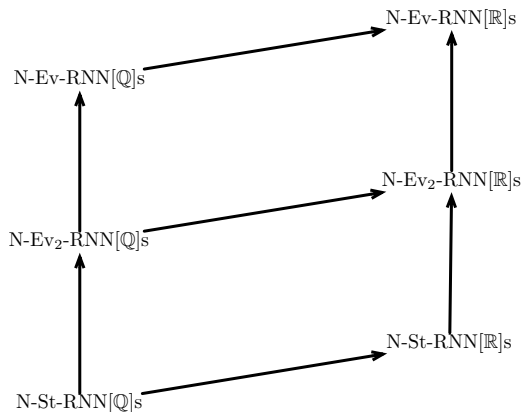
1. static rational RNNs: N-St-RNN[\mathbb{Q}]s
2. static real RNNs: N-St-RNN[\mathbb{R}]s
3. bi-valued evolving rational RNNs: N-Ev₂-RNN[\mathbb{Q}]s
4. bi-valued evolving real RNNs: N-Ev₂-RNN[\mathbb{R}]s
5. general evolving rational RNNs: N-Ev-RNN[\mathbb{Q}]s
6. general evolving real N-RNNs: N-Ev-RNN[\mathbb{R}]s



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

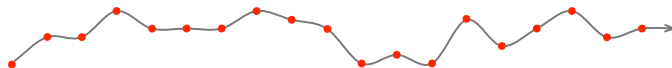
NONDETERMINISM OF TYPE I

Relationships between those models:

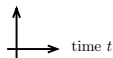


NONDETERMINISM OF TYPE II

A possible **evolution** $e = \vec{w}(0) \vec{w}(1) \vec{w}(2) \cdots \in \mathbb{Q}^K$ or \mathbb{R}^K

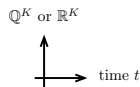
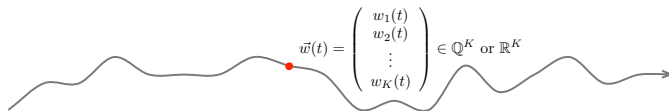


\mathbb{Q}^K or \mathbb{R}^K



NONDETERMINISM OF TYPE II

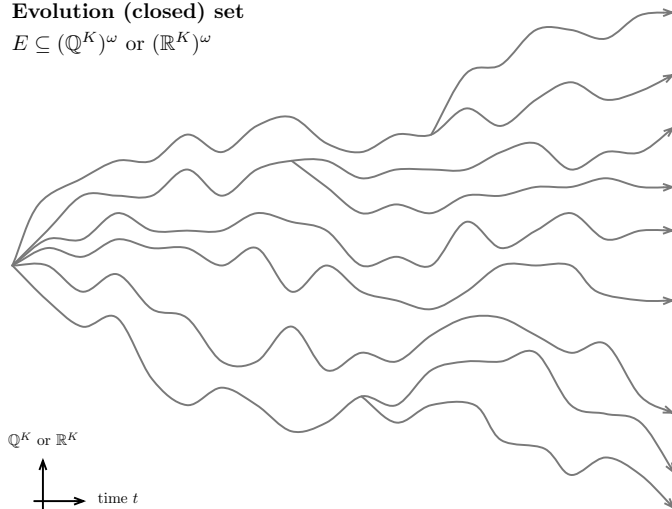
A possible **evolution** $e = \vec{w}(0) \vec{w}(1) \vec{w}(2) \dots \in \mathbb{Q}^K$ or \mathbb{R}^K



NONDETERMINISM OF TYPE II

Evolution (closed) set

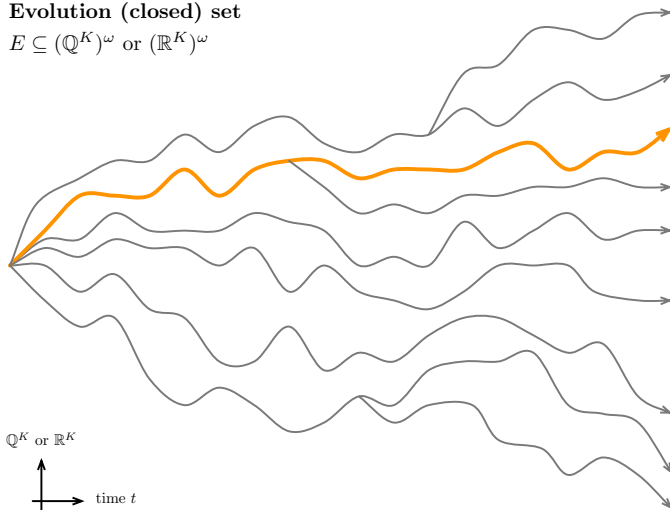
$$E \subseteq (\mathbb{Q}^K)^\omega \text{ or } (\mathbb{R}^K)^\omega$$



NONDETERMINISM OF TYPE II

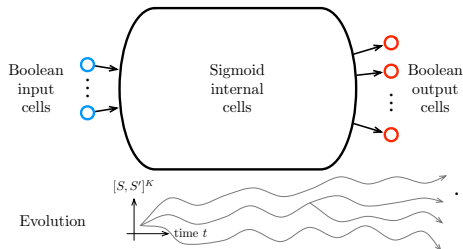
Evolution (closed) set

$$E \subseteq (\mathbb{Q}^K)^\omega \text{ or } (\mathbb{R}^K)^\omega$$



NONDETERMINISM OF TYPE II

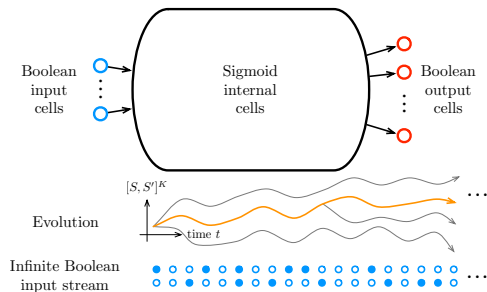
The RNNs are provided with an additional evolution set.



- Input stream $s \in (\mathbb{B}^M)^\omega$ accepted by \mathcal{N} iff there exists some evolution $e \in E$ s.t. $\mathcal{N}(s, e)$ enters a meaningful attractor.

NONDETERMINISM OF TYPE II

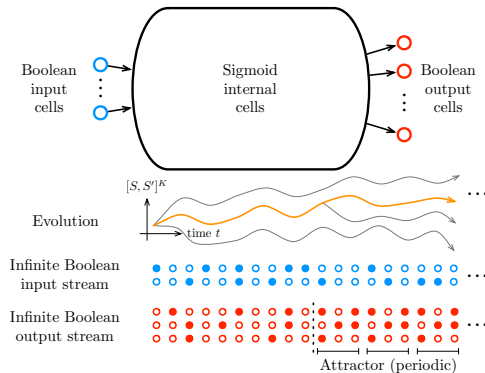
The RNNs are provided with an additional evolution set.



- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by \mathcal{N} iff there exists some evolution $e \in E$ s.t. $\mathcal{N}(s, e)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *rejected* by \mathcal{N} otherwise.

NONDETERMINISM OF TYPE II

The RNNs are provided with an additional evolution set.



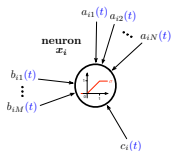
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by \mathcal{N} iff there exists some evolution $e \in E$ s.t. $\mathcal{N}(s, e)$ enters a meaningful attractor.
- ▶ Input stream $s \in (\mathbb{B}^M)^\omega$ *rejected* by \mathcal{N} otherwise.

NONDETERMINISTIC ω -RNNs: TYPE II

1. bi-valued evolving rational RNNs: $\tilde{N}\text{-Ev}_2\text{-RNN}[\mathbb{Q}]_s$
2. bi-valued evolving real RNNs: $\tilde{N}\text{-Ev}_2\text{-RNN}[\mathbb{R}]_s$
3. general evolving rational RNNs: $\tilde{N}\text{-Ev-RNN}[\mathbb{Q}]_s$
4. general evolving real N-RNNs: $\tilde{N}\text{-Ev-RNN}[\mathbb{R}]_s$

NONDETERMINISTIC ω -RNNs: TYPE II

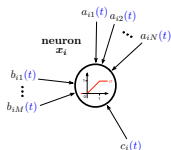
1. bi-valued evolving rational RNNs: $\tilde{\mathbf{N}}\text{-Ev}_2\text{-RNN}[\mathbb{Q}]_s$
2. bi-valued evolving real RNNs: $\tilde{\mathbf{N}}\text{-Ev}_2\text{-RNN}[\mathbb{R}]_s$
3. general evolving rational RNNs: $\tilde{\mathbf{N}}\text{-Ev-RNN}[\mathbb{Q}]_s$
4. general evolving real N-RNNs: $\tilde{\mathbf{N}}\text{-Ev-RNN}[\mathbb{R}]_s$



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

NONDETERMINISTIC ω -RNNs: TYPE II

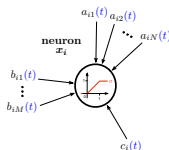
1. bi-valued evolving rational RNNs: $\tilde{\mathbf{N}}\text{-Ev}_2\text{-RNN}[\mathbb{Q}]_s$
2. bi-valued evolving real RNNs: $\tilde{\mathbf{N}}\text{-Ev}_2\text{-RNN}[\mathbb{R}]_s$
3. general evolving rational RNNs: $\tilde{\mathbf{N}}\text{-Ev-RNN}[\mathbb{Q}]_s$
4. general evolving real N-RNNs: $\tilde{\mathbf{N}}\text{-Ev-RNN}[\mathbb{R}]_s$



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

NONDETERMINISTIC ω -RNNs: TYPE II

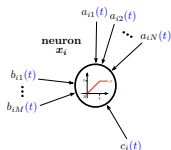
1. bi-valued evolving rational RNNs: $\tilde{N}\text{-Ev}_2\text{-RNN}[\mathbb{Q}]_s$
2. bi-valued evolving real RNNs: $\tilde{N}\text{-Ev}_2\text{-RNN}[\mathbb{R}]_s$
3. general evolving rational RNNs: $\tilde{N}\text{-Ev-RNN}[\mathbb{Q}]_s$
4. general evolving real N-RNNs: $\tilde{N}\text{-Ev-RNN}[\mathbb{R}]_s$



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

NONDETERMINISTIC ω -RNNs: TYPE II

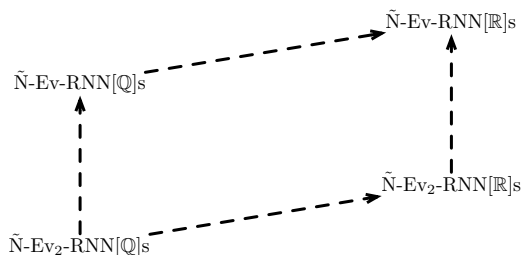
1. bi-valued evolving rational RNNs: $\tilde{N}\text{-Ev}_2\text{-RNN}[\mathbb{Q}]_s$
2. bi-valued evolving real RNNs: $\tilde{N}\text{-Ev}_2\text{-RNN}[\mathbb{R}]_s$
3. general evolving rational RNNs: $\tilde{N}\text{-Ev-RNN}[\mathbb{Q}]_s$
4. general evolving real N-RNNs: $\tilde{N}\text{-Ev-RNN}[\mathbb{R}]_s$



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

NONDETERMINISM OF TYPE II

Relationships between those models:



NONDETERMINISM OF TYPES I AND II

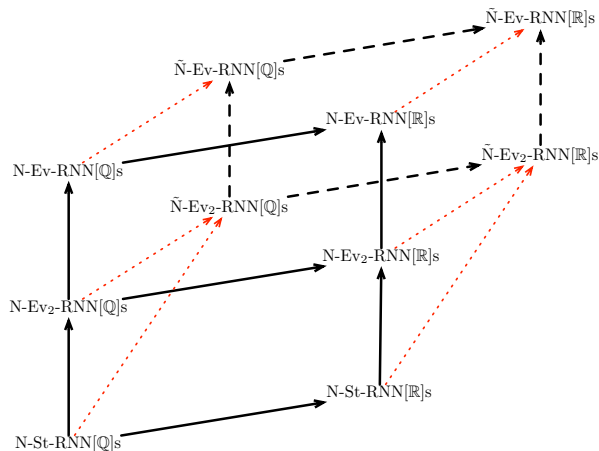
LEMMA

The nondeterminism of type I is a particular case of that of type II.

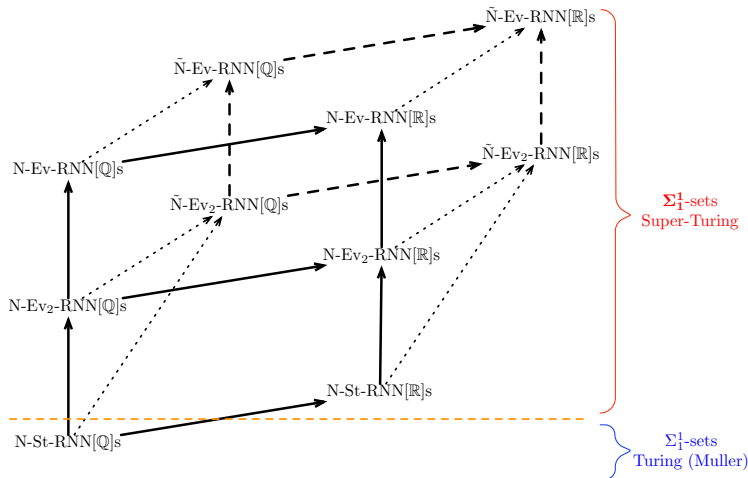
PROOF: simply build an evolution set that takes into account all possible guess streams.

NONDETERMINISM OF TYPES I AND II

Relationships between all nondeterministic models:



RESULTS



RESULTS

THEOREM

Let $L \subseteq (\mathbb{B}^M)^\omega$. The following conditions are equivalent.

- ▶ $L \in \Sigma_1^1$
- ▶ L is recognizable by some nondeterministic Muller TM
- ▶ L is recognizable by some N-St-RNN[\mathbb{Q}]

PROOF: Generalization of the classical equivalence between TMs and St-RNN[\mathbb{Q}] (Siegelmann & Sontag 95).

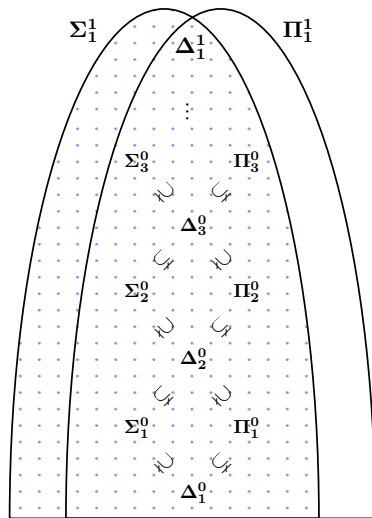
RESULTS

THEOREM

Let $L \subseteq (\mathbb{B}^M)^\omega$. The following conditions are equivalent.

- ▶ $L \in \Sigma_1^1$
- ▶ L is recognizable by some nondeterministic Muller TM
- ▶ L is recognizable by some N-St-RNN[\mathbb{Q}]

PROOF: Generalization of the classical equivalence between TMs and St-RNN[\mathbb{Q}] (Siegelmann & Sontag 95).



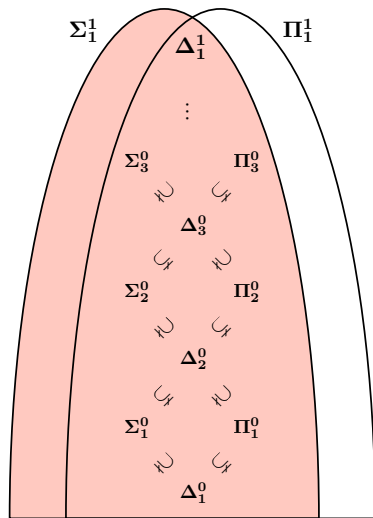
RESULTS

THEOREM

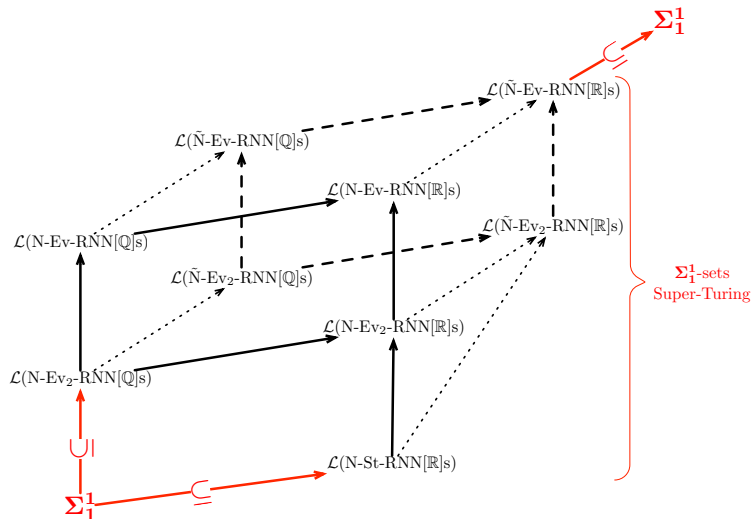
Let $L \subseteq (\mathbb{B}^M)^\omega$. The following conditions are equivalent.

- ▶ $L \in \Sigma_1^1$;
- ▶ L is recognizable by some N -St-RNN[\mathbb{R}];
- ▶ L is recognizable by some N -Ev₂-RNN[\mathbb{Q}];
- ▶ L is recognizable by some \tilde{N} -Ev₂-RNN[\mathbb{Q}];
- ▶ L is recognizable by some N -Ev-RNN[\mathbb{Q}];
- ▶ L is recognizable by some \tilde{N} -Ev-RNN[\mathbb{Q}];
- ▶ L is recognizable by some N -Ev₂-RNN[\mathbb{R}];
- ▶ L is recognizable by some \tilde{N} -Ev₂-RNN[\mathbb{R}];
- ▶ L is recognizable by some N -Ev-RNN[\mathbb{R}].
- ▶ L is recognizable by some \tilde{N} -Ev-RNN[\mathbb{R}].

RESULTS



RESULTS



RESULTS

PROPOSITION

Let $L \in \Sigma_1^1$. Then L is recognizable by some $N\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $N\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

➤ Since $L \in \Sigma_1^1$, there exists some Π_2^0 set $X \subseteq (\mathbb{B}^M)^\omega \times \{0,1\}^\omega$ such that $L = \pi_1(X)$.

$$X = \bigcap_{i \geq 0} \bigcup_{j \geq 0} (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0,1\}^\omega)$$

➤ X can be recursively encoded into some $r_X \in \mathbb{R}$.

RESULTS

PROPOSITION

Let $L \in \Sigma_1^1$. Then L is recognizable by some $N\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $N\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

- Since $L \in \Sigma_1^1$, there exists some Π_2^0 set $X \subseteq (\mathbb{B}^M)^\omega \times \{0, 1\}^\omega$ such that $L = \pi_1(X)$.

$$X = \bigcap_{i \geq 0} \bigcup_{j \geq 0} (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega)$$

- X can be recursively encoded into some $r_X \in \mathbb{R}$.

RESULTS

PROPOSITION

Let $L \in \Sigma_1^1$. Then L is recognizable by some $N\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$ or by some $N\text{-St-RNN}[\mathbb{R}]$.

PROOF (SKETCH):

- ▶ Since $L \in \Sigma_1^1$, there exists some Π_2^0 set $X \subseteq (\mathbb{B}^M)^\omega \times \{0, 1\}^\omega$ such that $L = \pi_1(X)$.

$$X = \bigcap_{i \geq 0} \bigcup_{j \geq 0} (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega)$$

- ▶ X can be recursively encoded into some $r_X \in \mathbb{R}$.

Algorithm 2 Infinite procedure

guess stream $g = g(0)g(1)g(2) \cdots \in \mathbb{B}^\omega$, and real number r_X .

RESULTS

Algorithm 2 Infinite procedure

Require: Input stream $s = \mathbf{u}(0)\mathbf{u}(1)\mathbf{u}(2) \cdots \in (\mathbb{B}^M)^\omega$,
 guess stream $g = g(0)g(1)g(2) \cdots \in \mathbb{B}^\omega$, and real number r_X .

- 1: store each $\mathbf{u}(t) \in \mathbb{B}^M$ and $g(t) \in \{0, 1\}$ as they arrive
- 2: $i \leftarrow 0, j \leftarrow 0$
- 3: **loop**
- 4: decode $(p_{i,j}, q_{i,j})$ from r_X // recursive procedure if r_X is given
- 5: **if** $p_{i,j} \subseteq s[0:c]$ and $q_{i,j} \subseteq g[0:c]$ **then** // $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 6: **return** 1 // $\exists j$ s.t. $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 7: $i \leftarrow i + 1, j \leftarrow 0$ // test if $(s, g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0, 1\}^\omega$
- 8: **else** // $(s, g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 9: **return** 0 // $\neg \exists j' \leq j$ s.t. $(s, g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0, 1\}^\omega$
- 10: $i \leftarrow i, j \leftarrow j + 1$ // test if $(s, g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0, 1\}^\omega$
- 11: **end if**
- 12: **end loop**

Algo returns ∞ -many 1's iff $(s, g) \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega) = X$.

RESULTS

Algorithm 2 Infinite procedure

Require: Input stream $s = u(0)u(1)u(2) \cdots \in (\mathbb{B}^M)^\omega$,
guess stream $g = g(0)g(1)g(2) \cdots \in \mathbb{B}^\omega$, and real number r_X .

- 1: store each $\mathbf{u}(t) \in \mathbb{B}^M$ and $g(t) \in \{0, 1\}$ as they arrive

2: $i \leftarrow 0, j \leftarrow 0$

RESULTS

Algorithm 2 Infinite procedure

Require: Input stream $s = u(0)u(1)u(2) \dots \in (\mathbb{B}^M)^\omega$,
 guess stream $g = g(0)g(1)g(2) \dots \in \mathbb{B}^\omega$, and real number r_X .

- 1: store each $u(t) \in \mathbb{B}^M$ and $g(t) \in \{0, 1\}$ as they arrive
- 2: $i \leftarrow 0, j \leftarrow 0$
- 3: **loop**
- 4: decode $(p_{i,j}, q_{i,j})$ from r_X // recursive procedure if r_X is given
- 5: **if** $p_{i,j} \subseteq s[0:c]$ and $q_{i,j} \subseteq g[0:c]$ **then** // $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 6: **return** 1 // $\exists j$ s.t. $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 7: $i \leftarrow i + 1, j \leftarrow 0$ // test if $(s, g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0, 1\}^\omega$
- 8: **else** // $(s, g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 9: **return** 0 // $\neg \exists j' \leq j$ s.t. $(s, g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0, 1\}^\omega$
- 10: $i \leftarrow i, j \leftarrow j + 1$ // test if $(s, g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0, 1\}^\omega$
- 11: **end if**
- 12: **end loop**

Algo returns ∞ -many 1's iff $(s, g) \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega) = X$.

RESULTS

Algorithm 2 Infinite procedure

Require: Input stream $s = u(0)u(1)u(2) \dots \in (\mathbb{B}^M)^\omega$,
 guess stream $g = g(0)g(1)g(2) \dots \in \mathbb{B}^\omega$, and real number r_X .

- 1: store each $u(t) \in \mathbb{B}^M$ and $g(t) \in \{0, 1\}$ as they arrive
- 2: $i \leftarrow 0, j \leftarrow 0$
- 3: **loop**
- 4: decode $(p_{i,j}, q_{i,j})$ from r_X // recursive procedure if r_X is given
- 5: **if** $p_{i,j} \subseteq s[0:c]$ and $q_{i,j} \subseteq g[0:c]$ **then** // $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 6: **return** 1 // $\exists j$ s.t. $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 7: $i \leftarrow i + 1, j \leftarrow 0$ // test if $(s, g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0, 1\}^\omega$
- 8: **else** // $(s, g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 9: **return** 0 // $\neg \exists j' \leq j$ s.t. $(s, g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0, 1\}^\omega$
- 10: $i \leftarrow i, j \leftarrow j + 1$ // test if $(s, g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0, 1\}^\omega$
- 11: **end if**
- 12: **end loop**

Algo returns ∞ -many 1's iff $(s, g) \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega) = X$.

RESULTS

Algorithm 2 Infinite procedure

Require: Input stream $s = u(0)u(1)u(2) \dots \in (\mathbb{B}^M)^\omega$,
 guess stream $g = g(0)g(1)g(2) \dots \in \mathbb{B}^\omega$, and real number r_X .

- 1: store each $u(t) \in \mathbb{B}^M$ and $g(t) \in \{0, 1\}$ as they arrive
- 2: $i \leftarrow 0, j \leftarrow 0$
- 3: **loop**
- 4: decode $(p_{i,j}, q_{i,j})$ from r_X // recursive procedure if r_X is given
- 5: **if** $p_{i,j} \subseteq s[0:c]$ and $q_{i,j} \subseteq g[0:c]$ **then** // $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 6: **return** 1 // $\exists j$ s.t. $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 7: $i \leftarrow i + 1, j \leftarrow 0$ // test if $(s, g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0, 1\}^\omega$
- 8: **else** // $(s, g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 9: **return** 0 // $\neg \exists j' \leq j$ s.t. $(s, g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0, 1\}^\omega$
- 10: $i \leftarrow i, j \leftarrow j + 1$ // test if $(s, g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0, 1\}^\omega$
- 11: **end if**
- 12: **end loop**

Algo returns ∞ -many 1's iff $(s, g) \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega) = X$.

RESULTS

Algorithm 2 Infinite procedure

Require: Input stream $s = u(0)u(1)u(2) \dots \in (\mathbb{B}^M)^\omega$,
 guess stream $g = g(0)g(1)g(2) \dots \in \mathbb{B}^\omega$, and real number r_X .

- 1: store each $u(t) \in \mathbb{B}^M$ and $g(t) \in \{0, 1\}$ as they arrive
- 2: $i \leftarrow 0, j \leftarrow 0$
- 3: **loop**
- 4: decode $(p_{i,j}, q_{i,j})$ from r_X // recursive procedure if r_X is given
- 5: **if** $p_{i,j} \subseteq s[0:c]$ and $q_{i,j} \subseteq g[0:c]$ **then** // $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 6: **return** 1 // $\exists j$ s.t. $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 7: $i \leftarrow i + 1, j \leftarrow 0$ // test if $(s, g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0, 1\}^\omega$
- 8: **else** // $(s, g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 9: **return** 0 // $\neg \exists j' \leq j$ s.t. $(s, g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0, 1\}^\omega$
- 10: $i \leftarrow i, j \leftarrow j + 1$ // test if $(s, g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0, 1\}^\omega$
- 11: **end if**
- 12: **end loop**

Algo returns ∞ -many 1's iff $(s, g) \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega) = X$.

RESULTS

Algorithm 2 Infinite procedure

Require: Input stream $s = u(0)u(1)u(2) \dots \in (\mathbb{B}^M)^\omega$,
 guess stream $g = g(0)g(1)g(2) \dots \in \mathbb{B}^\omega$, and real number r_X .

- 1: store each $u(t) \in \mathbb{B}^M$ and $g(t) \in \{0, 1\}$ as they arrive
- 2: $i \leftarrow 0, j \leftarrow 0$
- 3: **loop**
- 4: decode $(p_{i,j}, q_{i,j})$ from r_X // recursive procedure if r_X is given
- 5: **if** $p_{i,j} \subseteq s[0:c]$ and $q_{i,j} \subseteq g[0:c]$ **then** // $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 6: **return** 1 // $\exists j$ s.t. $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 7: $i \leftarrow i + 1, j \leftarrow 0$ // test if $(s, g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0, 1\}^\omega$
- 8: **else** // $(s, g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$
- 9: **return** 0 // $\neg \exists j' \leq j$ s.t. $(s, g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0, 1\}^\omega$
- 10: $i \leftarrow i, j \leftarrow j + 1$ // test if $(s, g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0, 1\}^\omega$
- 11: **end if**
- 12: **end loop**

Algo returns ∞ -many 1's iff $(s, g) \in \bigcap_i \bigcup_j (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega) = X$.

Algorithm

```

Require: Input stream  $s = \mathbf{u}(0)\mathbf{u}(1)\mathbf{u}(2) \cdots \in (\mathbb{B}^M)^\omega$ ,
guess stream  $g = g(0)g(1)g(2) \cdots \in \mathbb{B}^\omega$ , and real number  $r_X$ .

1: store each  $\mathbf{u}(t) \in \mathbb{B}^M$  and  $g(t) \in \{0, 1\}$  as they arrive
2:  $i \leftarrow 0, j \leftarrow 0$ 
3: loop
4:   decode  $(p_{i,j}, q_{i,j})$  from  $r_X$  // recursive procedure if  $r_X$  is given
5:   if  $p_{i,j} \subseteq s[0:c]$  and  $q_{i,j} \subseteq g[0:c]$  then //  $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$ 
6:     return 1 //  $\exists j$  s.t.  $(s, g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$ 
7:      $i \leftarrow i + 1, j \leftarrow 0$  // test if  $(s, g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0, 1\}^\omega$ 
8:   else //  $(s, g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0, 1\}^\omega$ 
9:     return 0 //  $\neg \exists j' \leq j$  s.t.  $(s, g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0, 1\}^\omega$ 
10:     $i \leftarrow i, j \leftarrow j + 1$  // test if  $(s, g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0, 1\}^\omega$ 
11:  end if
12: end loop

```

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

JÉRÉMIE CABESSA

RESULTS

- ▶ The algo can be simulated by some N-St-RNN[\mathbb{R}] \mathcal{N} , i.e., algo returns infinitely many 1's on (s, g) iff $\mathcal{N}(s, g)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{R}^n$ s.t. $\mathcal{N}(s, g)$ visits a meaningful attractor
 - ▶ iff $\exists g \in \mathbb{R}^n$ s.t. the algorithm returns 1 on the word (s, g)
 - ▶ iff $\exists g \in \mathbb{R}^n$ s.t. $(s, g) \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{R}^n$ s.t. $(s, g) \in L$
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some N-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on (s, g) iff $\mathcal{N}(s, g)$ visits a
 meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ visits a meaningful attractor
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. the algo returns infinitely many 1's on (s, g)
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $(s, g) \in X$
 - ▶ iff $s \in \pi_1(X) = L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some N-St-RNN[\mathbb{R}] \mathcal{N} , i.e., algo returns infinitely many 1's on (s, g) iff $\mathcal{N}(s, g)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ visits a meaningful attractor
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. the algo returns infinitely many 1's on (s, g)
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $(s, g) \in X$
 - ▶ iff $s \in \pi_1(X) = L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some N-St-RNN[\mathbb{R}] \mathcal{N} , i.e., algo returns infinitely many 1's on (s, g) iff $\mathcal{N}(s, g)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ visits a meaningful attractor
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. the algo returns infinitely many 1's on (s, g)
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $(s, g) \in X$
 - ▶ iff $s \in \pi_1(X) = L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some N-St-RNN[\mathbb{R}] \mathcal{N} , i.e., algo returns infinitely many 1's on (s, g) iff $\mathcal{N}(s, g)$ visits a meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ visits a meaningful attractor
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. the algo returns infinitely many 1's on (s, g)
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $(s, g) \in X$
 - ▶ iff $s \in \pi_1(X) = L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some N-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on (s, g) iff $\mathcal{N}(s, g)$ visits a
 meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ visits a meaningful attractor
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. the algo returns infinitely many 1's on (s, g)
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $(s, g) \in X$
 - ▶ iff $s \in \pi_1(X) = L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

- ▶ The algo can be simulated by some N-St-RNN[\mathbb{R}] \mathcal{N} , i.e.,
 algo returns infinitely many 1's on (s, g) iff $\mathcal{N}(s, g)$ visits a
 meaningful attractor.
- ▶ $s \in L(\mathcal{N})$
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $\mathcal{N}(s, g)$ visits a meaningful attractor
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. the algo returns infinitely many 1's on (s, g)
 - ▶ iff $\exists g \in \mathbb{B}^\omega$ s.t. $(s, g) \in X$
 - ▶ iff $s \in \pi_1(X) = L$.
- ▶ Therefore, $L(\mathcal{N}) = L$. □

RESULTS

PROPOSITION

Let \mathcal{N} be some \tilde{N} -Ev-RNN $[\mathbb{R}]$. Then $L(\mathcal{N}) \in \Sigma_1^1$.

PROOF (SKETCH):

- ▶ The function $f_{\mathcal{N}} : (\mathbb{B}^M)^\omega \times E \rightarrow (\mathbb{B}^P)^\omega$ associated with the dynamics of \mathcal{N} is of Baire class 1 (preimage of a Σ_1^0 is Σ_2^0).
- ▶ Accordingly, the ω -language $L(\mathcal{N})$ can be expressed as the first projection of a finite Boolean combination of Σ_3^0 and Π_3^0 sets (i.e., of a Borel set) of the Polish space $(\mathbb{B}^M)^\omega \times E$.
- ▶ Therefore, $L(\mathcal{N}) \in \Sigma_1^1$. □

RESULTS

PROPOSITION

Let \mathcal{N} be some \tilde{N} -Ev-RNN $[\mathbb{R}]$. Then $L(\mathcal{N}) \in \Sigma_1^1$.

PROOF (SKETCH):

- ▶ The function $f_{\mathcal{N}} : (\mathbb{B}^M)^\omega \times E \rightarrow (\mathbb{B}^P)^\omega$ associated with the dynamics of \mathcal{N} is of Baire class 1 (preimage of a Σ_1^0 is Σ_2^0).
- ▶ Accordingly, the ω -language $L(\mathcal{N})$ can be expressed as the first projection of a finite Boolean combination of Σ_3^0 and Π_3^0 sets (i.e., of a Borel set) of the Polish space $(\mathbb{B}^M)^\omega \times E$.
- ▶ Therefore, $L(\mathcal{N}) \in \Sigma_1^1$. □

RESULTS – SUMMARY

NONDET.	STATIC	BI-VALUED EVOLVING	GENERAL EVOLVING
\mathbb{Q}	N-St-RNN[\mathbb{Q}]s – $= \Sigma_1^1$ (lightface) Turing (Muller)	N-Ev ₂ -RNN[\mathbb{Q}]s $\tilde{\text{N}}$ -Ev ₂ -RNN[\mathbb{Q}]s $= \Sigma_1^1$ (boldface) super-Turing	N-Ev-RNN[\mathbb{Q}]s $\tilde{\text{N}}$ -Ev-RNN[\mathbb{Q}]s $= \Sigma_1^1$ (boldface) super-Turing
\mathbb{R}	N-St-RNN[\mathbb{R}]s – $= \Sigma_1^1$ (boldface) super-Turing	N-Ev ₂ -RNN[\mathbb{R}]s $\tilde{\text{N}}$ -Ev ₂ -RNN[\mathbb{R}]s $= \Sigma_1^1$ (boldface) super-Turing	N-Ev-RNN[\mathbb{R}]s $\tilde{\text{N}}$ -Ev-RNN[\mathbb{R}]s $= \Sigma_1^1$ (boldface) super-Turing

CONCLUSION

- ▶ The issue of *computational power* precedes that of *learning*.
- ▶ We provided a characterization of the expressive power of recurrent neural networks working on infinite inputs.
- ▶ In general, the super-Turing computational capabilities of neural models raises the question of *hypercomputation*.
- ▶ Current physical theories are consistent with the possibility of hypercomputational systems (quantum, relativistic, etc.). No such systems are currently feasible or harnessable.

CONCLUSION

- ▶ The issue of *computational power* precedes that of *learning*.
- ▶ We provided a characterization of the expressive power of recurrent neural networks working on infinite inputs.
- ▶ In general, the super-Turing computational capabilities of neural models raises the question of *hypercomputation*.
- ▶ Current physical theories are consistent with the possibility of hypercomputational systems (quantum, relativistic, etc.). No such systems are currently feasible or harnessable.

CONCLUSION

- ▶ The issue of *computational power* precedes that of *learning*.
- ▶ We provided a characterization of the expressive power of recurrent neural networks working on infinite inputs.
- ▶ In general, the super-Turing computational capabilities of neural models raises the question of *hypercomputation*.
- ▶ Current physical theories are consistent with the possibility of hypercomputational systems (quantum, relativistic, etc.). No such systems are currently feasible or harnessable.

CONCLUSION

- ▶ The issue of *computational power* precedes that of *learning*.
- ▶ We provided a characterization of the expressive power of recurrent neural networks working on infinite inputs.
- ▶ In general, the super-Turing computational capabilities of neural models raises the question of *hypercomputation*.
- ▶ Current physical theories are consistent with the possibility of hypercomputational systems (quantum, relativistic, etc.). No such systems are currently feasible or harnessable.