Part 1: Computational Capabilities of Recurrent Neural Networks

Jérémie Cabessa

Laboratoire d'Économie Mathématique (LEMMA)
Université Paris II

4 October 2016

Introduction

- We assume that some aspects of information processing in the brain can be approached from the perspective of computability theory.
- We consider neural network models involved in various (bio-inspired) computational paradigms.
- We analyze their computational capabilities...

Introduction

- We assume that some aspects of information processing in the brain can be approached from the perspective of computability theory.
- We consider neural network models involved in various (bio-inspired) computational paradigms.
- We analyze their computational capabilities...

Introduction

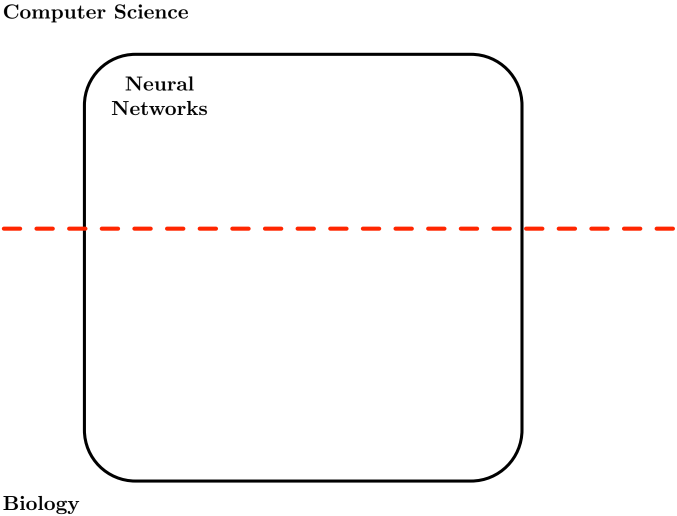
- We assume that some aspects of information processing in the brain can be approached from the perspective of computability theory.
- We consider neural network models involved in various (bio-inspired) computational paradigms.
- We analyze their computational capabilities...

Introduction

Computer Science

Biology

Introduction



Introduction

Computer Science

Neural Networks

Artificial Neural Networks

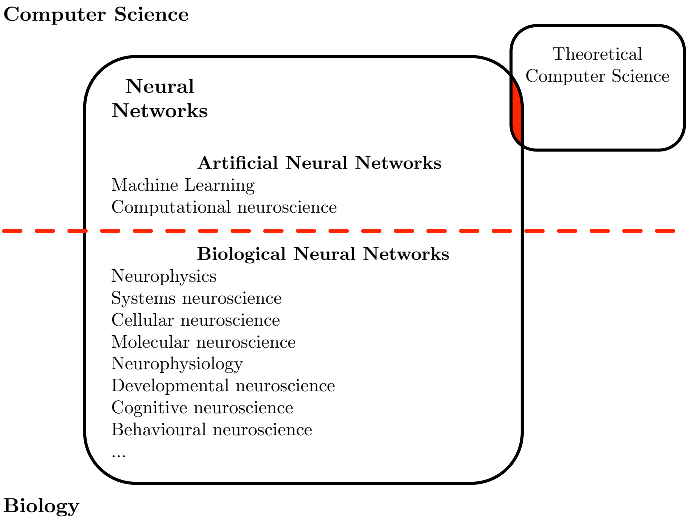
Machine Learning
Computational neuroscience

Biological Neural Networks

Neurophysics
Systems neuroscience
Cellular neuroscience
Molecular neuroscience
Neurophysiology
Developmental neuroscience
Cognitive neuroscience
Behavioural neuroscience
...

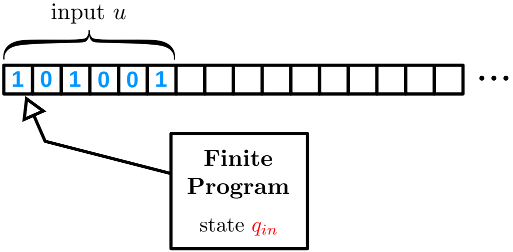
Biology

Introduction



Turing Machine

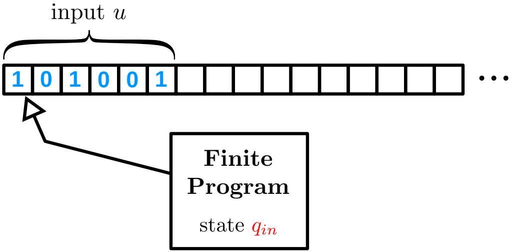
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

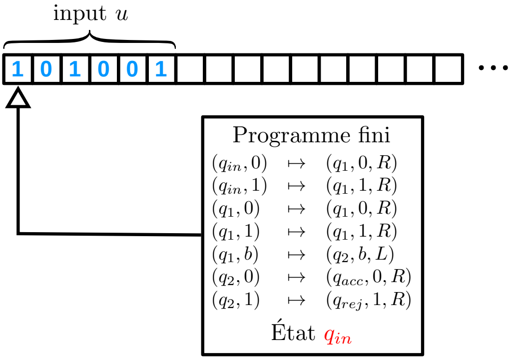
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

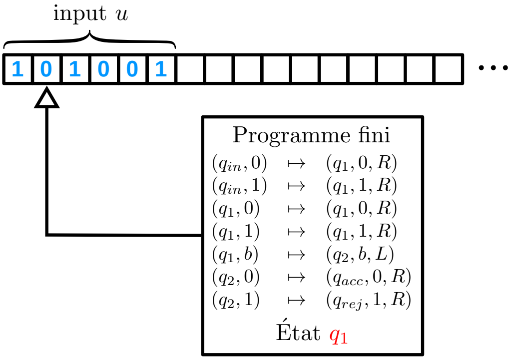
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

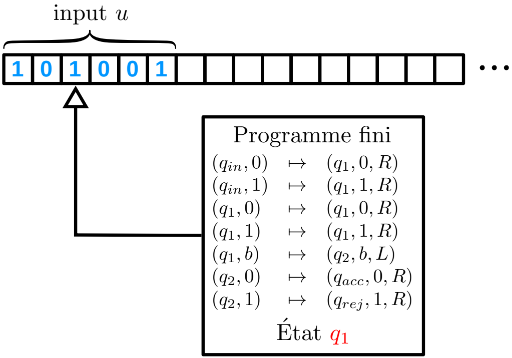
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

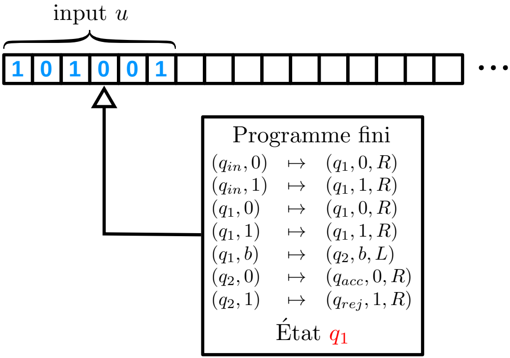
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

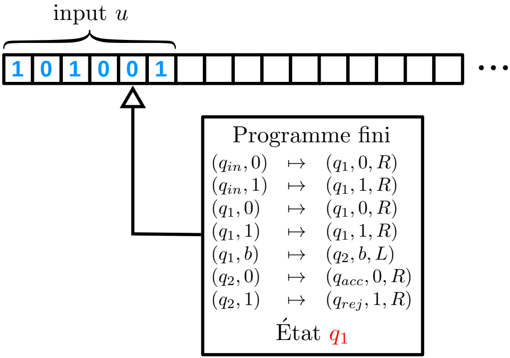
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

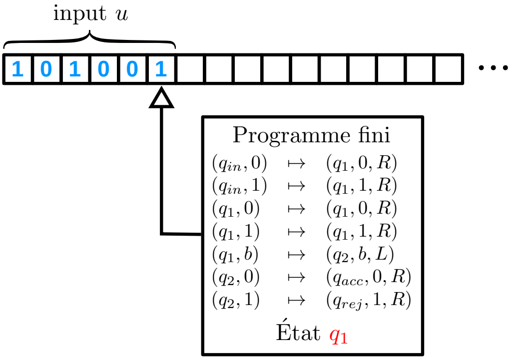
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

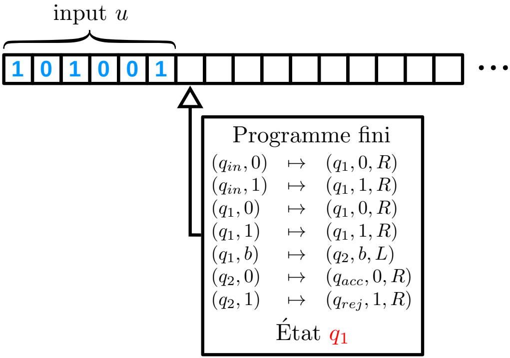
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

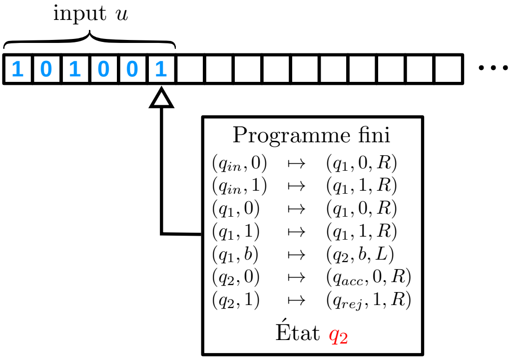
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

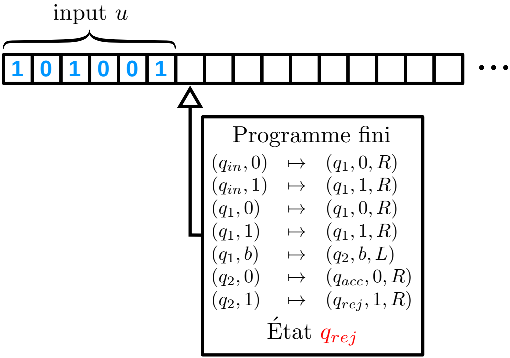
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- 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

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}

Church-Turing Thesis

A function is effectively computable if and only if it is Turing-computable.

- Informal statement setting the limits of effective computation.
- Implications in philosophy of mind, theoretical psychology, cognitive science, Artificial Intelligence, and Artificial Life.
- Sometimes under debate...

Church-Turing Thesis

A function is effectively computable if and only if it is Turing-computable.

- Informal statement setting the limits of effective computation.
- Implications in philosophy of mind, theoretical psychology, cognitive science, Artificial Intelligence, and Artificial Life.
- Sometimes under debate...

Church-Turing Thesis

A function is effectively computable if and only if it is Turing-computable.

- Informal statement setting the limits of effective computation.
- Implications in philosophy of mind, theoretical psychology, cognitive science, Artificial Intelligence, and Artificial Life.
- Sometimes under debate...

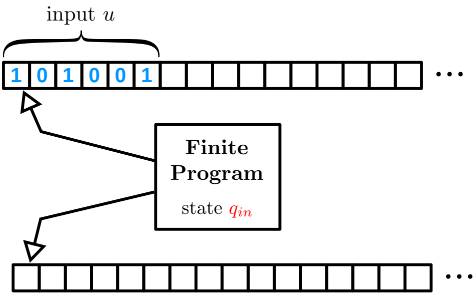
Church-Turing Thesis

A function is effectively computable if and only if it is Turing-computable.

- Informal statement setting the limits of effective computation.
- Implications in philosophy of mind, theoretical psychology, cognitive science, Artificial Intelligence, and Artificial Life.
- Sometimes under debate...

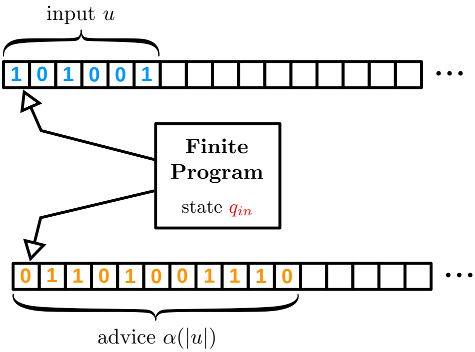
Turing Machine with Advice

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



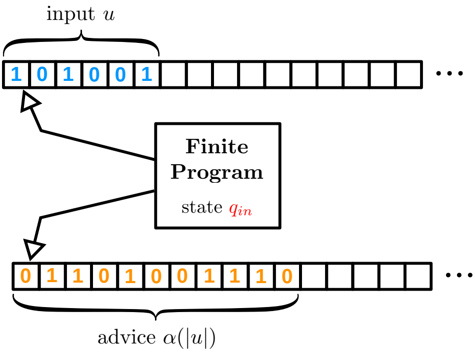
Turing Machine with Advice

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



Turing Machine with Advice

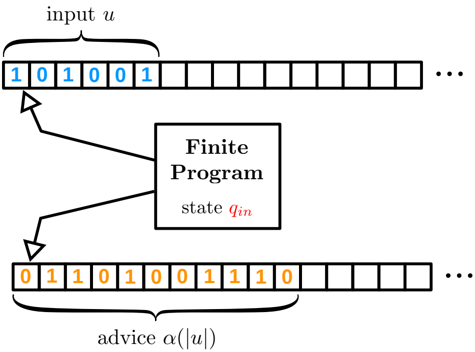
A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



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

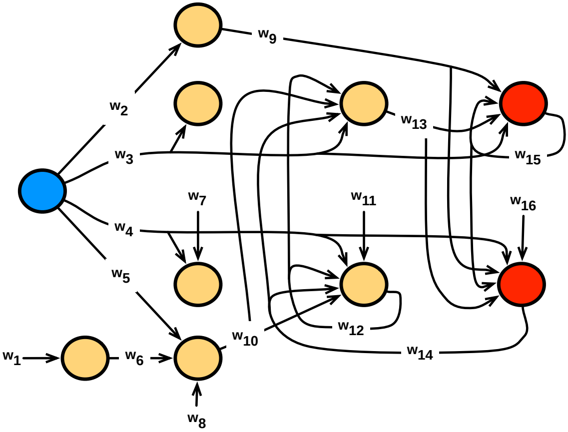
Turing Machine with Advice

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

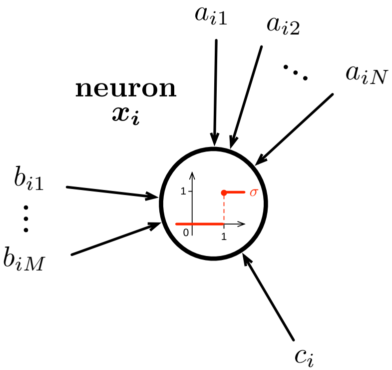


- A TM/A si strictly more powerful than a TM...
  We call this *super-Turing*.

Recurrent Neural Network

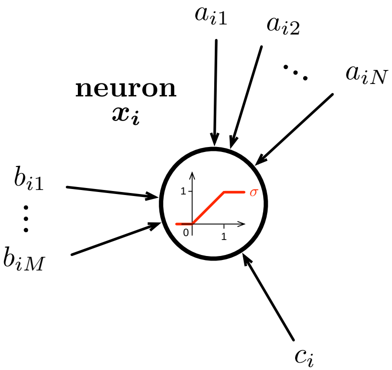


Boolean Recurrent Neural Networks



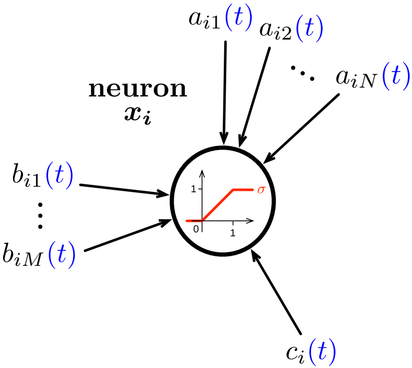
$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

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



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

Neural Activity

Video of firing neurons...

Link

Recurrent Neural Networks

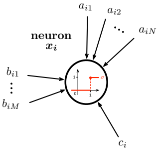
We consider eight models of RNNs:

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

Recurrent Neural Networks

We consider eight models of RNNs:

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

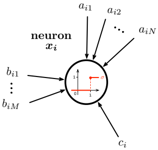


$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

Recurrent Neural Networks

We consider eight models of RNNs:

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

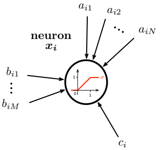


$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

Recurrent Neural Networks

We consider eight models of RNNs:

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

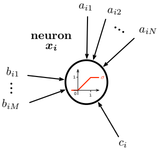


$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

Recurrent Neural Networks

We consider eight models of RNNs:

1. Boolean rational RNNs: B-RNN[ℚ]s
2. Boolean real RNNs: B-RNN[ℝ]s
3. Sigmoidal static rational RNNs: St-RNN[ℚ]s
4. Sigmoidal static real RNNs: St-RNN[ℝ]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev_2-RNN[ℚ]s
6. Sigmoidal bi-valued evolving real RNNs: Ev_2-RNN[ℝ]s
7. Sigmoidal general evolving rational RNNs: Ev-RNN[ℚ]s
8. Sigmoidal general evolving real N-RNNs: Ev-RNN[ℝ]s

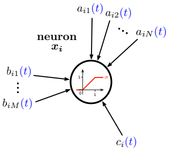


$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

Recurrent Neural Networks

We consider eight models of RNNs:

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

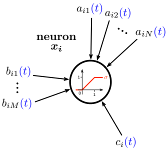


$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t)\cdot u_j(t) + c_i(t)\right)$$

Recurrent Neural Networks

We consider eight models of RNNs:

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

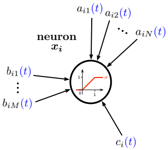


$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t)\cdot u_j(t) + c_i(t)\right)$$

Recurrent Neural Networks

We consider eight models of RNNs:

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

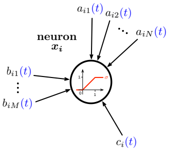


$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t) \cdot u_j(t) + c_i(t)\right)$$

Recurrent Neural Networks

We consider eight models of RNNs:

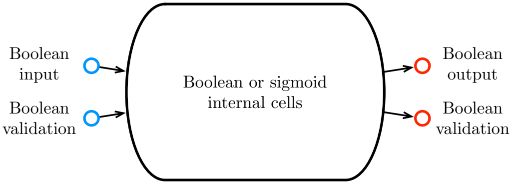
1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoidal static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoidal static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoidal bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoidal bi-valued evolving 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



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

RNNs as Language Recognizers

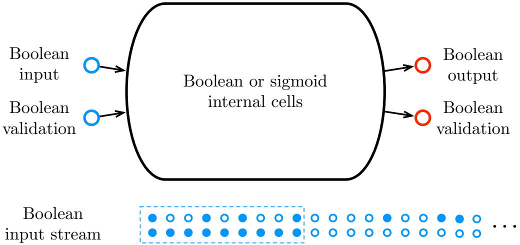
We consider RNNs with Boolean input and output cells, Boolean or sigmoidal internal cells.



- Input stream $s \in \mathbb{B}^*$ *rejected* by $\mathcal{N}$ iff $\mathcal{N}(s) = 0$.
- Input stream $s \in \mathbb{B}^*$ *accepted* by $\mathcal{N}$ iff $\mathcal{N}(s) = 1$.

RNNs as Language Recognizers

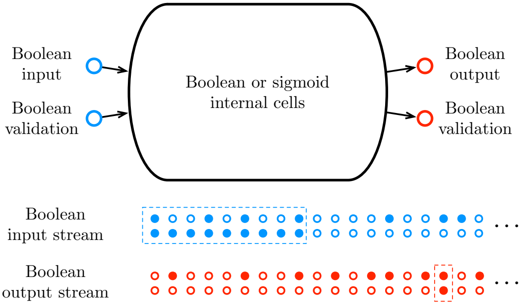
We consider RNNs with Boolean input and output cells, Boolean or sigmoidal internal cells.



- Input stream $s \in \mathbb{B}^*$ *rejected* by $\mathcal{N}$ iff $\mathcal{N}(s) = 0$.
- Input stream $s \in \mathbb{B}^*$ *accepted* by $\mathcal{N}$ iff $\mathcal{N}(s) = 1$.

RNNs as Language Recognizers

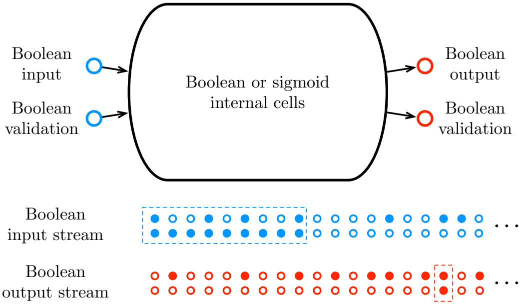
We consider RNNs with Boolean input and output cells, Boolean or sigmoidal internal cells.



- Input stream $s \in \mathbb{B}^*$ *rejected* by $\mathcal{N}$ iff $\mathcal{N}(s) = 0$.
- Input stream $s \in \mathbb{B}^*$ *accepted* by $\mathcal{N}$ iff $\mathcal{N}(s) = 1$.

RNNs as Language Recognizers

We consider RNNs with Boolean input and output cells, Boolean or sigmoidal internal cells.



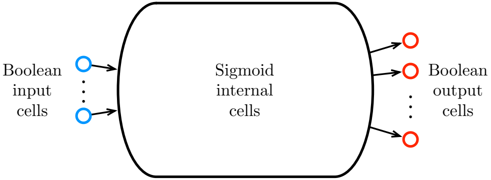
- Input stream $s \in \mathbb{B}^*$ *rejected* by $\mathcal{N}$ iff $\mathcal{N}(s) = 0$.
- Input stream $s \in \mathbb{B}^*$ *accepted* by $\mathcal{N}$ iff $\mathcal{N}(s) = 1$.

Results

	Boolean	Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA **REG** Kl 56, Mi 67	TM **P** Si & So 95	TM/poly(A) **P/poly** Ca & Si 11,14	TM/poly(A) **P/poly** Ca & Si 11,14
$\mathbb{R}$	FSA **REG** Kl 56, Mi 67	TM/poly(A) **P/poly** Si & So 94	TM/poly(A) **P/poly** Ca & Si 11,14	TM/poly(A) **P/poly** Ca & Si 11,14

Deterministic ω-RNNs

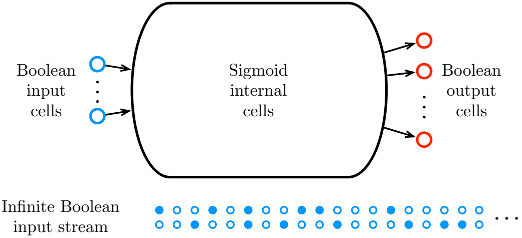
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite input streams.



- 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 $\mathcal{N}$ iff $\mathcal{N}(s)$ enters a spurious attractor.

Deterministic ω-RNNs

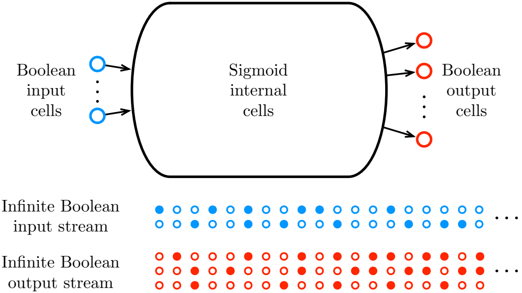
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite input streams.



- 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 $\mathcal{N}$ iff $\mathcal{N}(s)$ enters a spurious attractor.

Deterministic ω-RNNs

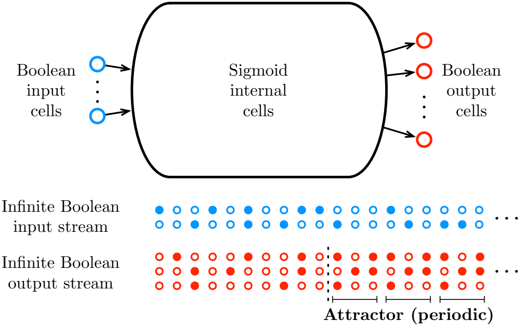
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite input streams.



- 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 $\mathcal{N}$ iff $\mathcal{N}(s)$ enters a spurious attractor.

Deterministic ω-RNNs

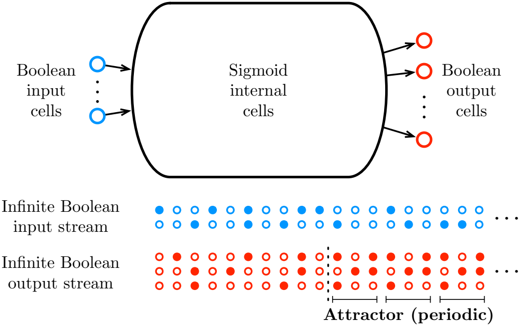
We consider RNNs with Boolean input and output cells, sigmoidal internal cells, and working on infinite input streams.



- 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 $\mathcal{N}$ iff $\mathcal{N}(s)$ enters a spurious attractor.

Deterministic ω-RNNs

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



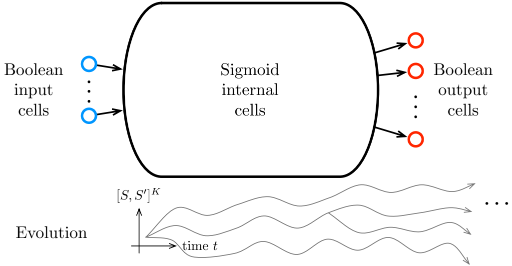
- Input stream $s \in (\mathbb{B}^M)^\omega$ *accepted* by $\mathcal{N}$ iff $\mathcal{N}(s)$ enters a meaningful attractor.
- Input stream $s \in (\mathbb{B}^M)^\omega$ *rejected* by $\mathcal{N}$ iff $\mathcal{N}(s)$ enters a spurious attractor.

Results

DET.	Static	Bi-valued Evolving	General Evolving
$\mathbb{Q}$	D-St-RNN[$\mathbb{Q}$]s $\in BC(\mathbf{\Pi^0_2})$ Turing (Muller)	D-Ev$_2$-RNN[$\mathbb{Q}$]s $= BC(\mathbf{\Pi^0_2})$ super-Turing	D-Ev-RNN[$\mathbb{Q}$]s $= BC(\mathbf{\Pi^0_2})$ super-Turing
$\mathbb{R}$	D-St-RNN[$\mathbb{R}$]s $= BC(\mathbf{\Pi^0_2})$ super-Turing	D-Ev$_2$-RNN[$\mathbb{R}$]s $= BC(\mathbf{\Pi^0_2})$ super-Turing	D-Ev-RNN[$\mathbb{R}$]s $= BC(\mathbf{\Pi^0_2})$ super-Turing

Nondeterministic ω-RNNs (Type II)

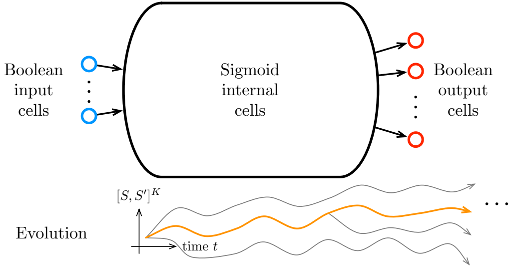
The RNNs are provided with an additional evolution set.



- 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 $\mathcal{N}$

Nondeterministic ω-RNNs (Type II)

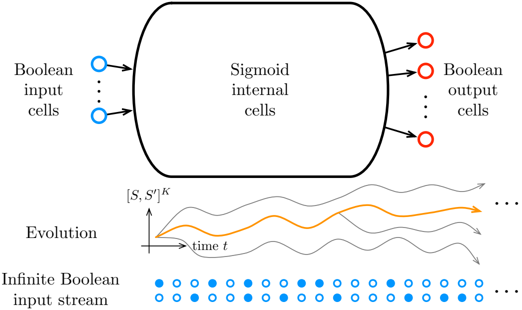
The RNNs are provided with an additional evolution set.



- 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)

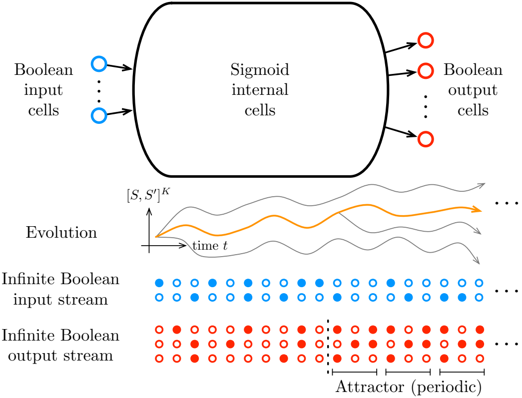
The RNNs are provided with an additional evolution set.



- 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)

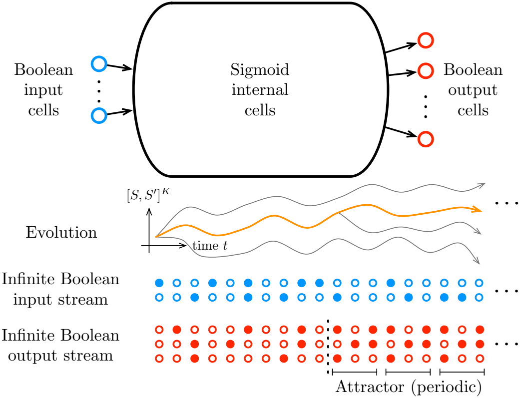
The RNNs are provided with an additional evolution set.



- 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)

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.

Results

NONDET.	Static	Bi-valued Evolving	General Evolving
$\mathbb{Q}$	N-St-RNN[$\mathbb{Q}$]s	N-Ev$_2$-RNN[$\mathbb{Q}$]s	N-Ev-RNN[$\mathbb{Q}$]s
	–	Ñ-Ev$_2$-RNN[$\mathbb{Q}$]s	Ñ-Ev-RNN[$\mathbb{Q}$]s
	$= \Sigma^1_1$ (lightface)	$= \boldsymbol{\Sigma}^1_1$ (boldface)	$= \boldsymbol{\Sigma}^1_1$ (boldface)
	Turing (Muller)	super-Turing	super-Turing
$\mathbb{R}$	N-St-RNN[$\mathbb{R}$]s	N-Ev$_2$-RNN[$\mathbb{R}$]s	N-Ev-RNN[$\mathbb{R}$]s
	–	Ñ-Ev$_2$-RNN[$\mathbb{R}$]s	Ñ-Ev-RNN[$\mathbb{R}$]s
	$= \boldsymbol{\Sigma}^1_1$ (boldface)	$= \boldsymbol{\Sigma}^1_1$ (boldface)	$= \boldsymbol{\Sigma}^1_1$ (boldface)
	super-Turing	super-Turing	super-Turing

Conclusions

- We provided a characterization of the expressive power of several models of recurrent neural networks involved in various computational paradigms.
- The power of the continuum (real synaptic weights) does add computational capabilities.
- The synaptic plasticity (evolving synaptic weights) add equivalent computational capabilities.
- Future work: super-Turing hierarchization in terms of the evolving speed of the networks.

Conclusions

- We provided a characterization of the expressive power of several models of recurrent neural networks involved in various computational paradigms.
- The power of the continuum (real synaptic weights) does add computational capabilities.
- The synaptic plasticity (evolving synaptic weights) add equivalent computational capabilities.
- Future work: super-Turing hierarchization in terms of the evolving speed of the networks.

Conclusions

- We provided a characterization of the expressive power of several models of recurrent neural networks involved in various computational paradigms.
- The power of the continuum (real synaptic weights) does add computational capabilities.
- The synaptic plasticity (evolving synaptic weights) add equivalent computational capabilities.
- Future work: super-Turing hierarchization in terms of the evolving speed of the networks.

Conclusions

- We provided a characterization of the expressive power of several models of recurrent neural networks involved in various computational paradigms.
- The power of the continuum (real synaptic weights) does add computational capabilities.
- The synaptic plasticity (evolving synaptic weights) add equivalent computational capabilities.
- Future work: super-Turing hierarchization in terms of the evolving speed of the networks.

Conclusions

- In general, the super-Turing computational capabilities of neural models raise the issue of *hypercomputation*.
- Current physical theories are consistent with the possibility of hypercomputational systems (e.g., quantum, relativistic). No such systems are currently feasible or harnessable.
- Philosophical considerations: What is computation? Extension of the Church-Turing Thesis; Thesis of Analog Computation (Siegelmann & Sontag [1995]); Church-Turing Thesis of Interactive Computation (van Leeuven & Wiedermann [2001]).

Conclusions

- In general, the super-Turing computational capabilities of neural models raise the issue of *hypercomputation*.
- Current physical theories are consistent with the possibility of hypercomputational systems (e.g., quantum, relativistic). No such systems are currently feasible or harnessable.
- Philosophical considerations: What is computation? Extension of the Church-Turing Thesis; Thesis of Analog Computation (Siegelmann & Sontag [1995]); Church-Turing Thesis of Interactive Computation (van Leeuven & Wiedermann [2001]).

Conclusions

- In general, the super-Turing computational capabilities of neural models raise the issue of *hypercomputation*.
- Current physical theories are consistent with the possibility of hypercomputational systems (e.g., quantum, relativistic). No such systems are currently feasible or harnessable.
- Philosophical considerations: What is computation? Extension of the Church-Turing Thesis; Thesis of Analog Computation (Siegelmann & Sontag [1995]); Church-Turing Thesis of Interactive Computation (van Leeuven & Wiedermann [2001]).

Part 2:
Limit Knowledge: A Topological Approach to Epistemic Game Theory

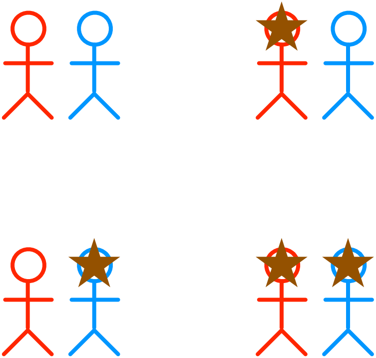
Jérémie Cabessa

Laboratoire d'Économie Mathématique (LEMMA)
Université Paris II

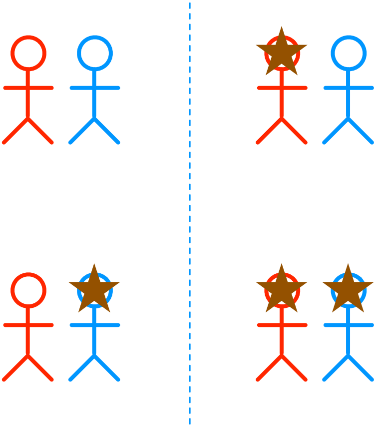
4 October 2016

Introduction: The Muddy Children Puzzle

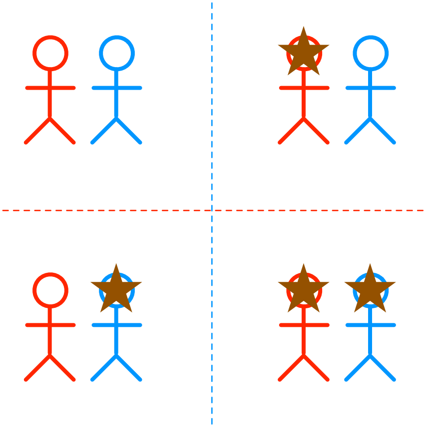
Introduction: The Muddy Children Puzzle



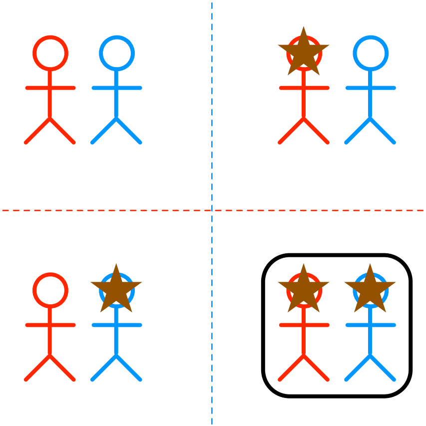
Introduction: The Muddy Children Puzzle



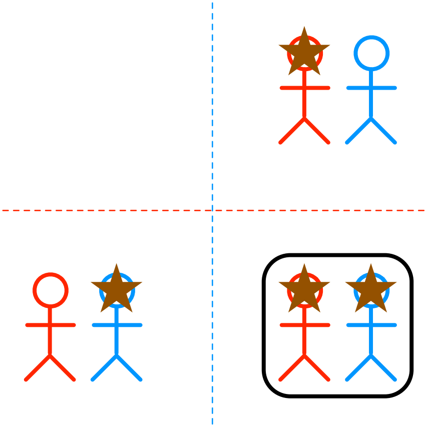
Introduction: The Muddy Children Puzzle



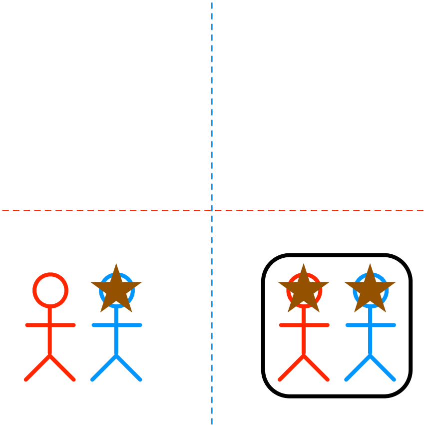
Introduction: The Muddy Children Puzzle



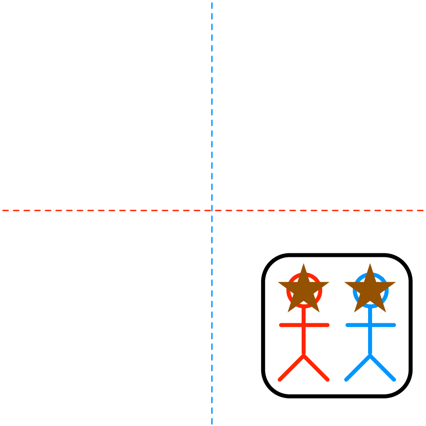
Introduction: The Muddy Children Puzzle



Introduction: The Muddy Children Puzzle



Introduction: The Muddy Children Puzzle



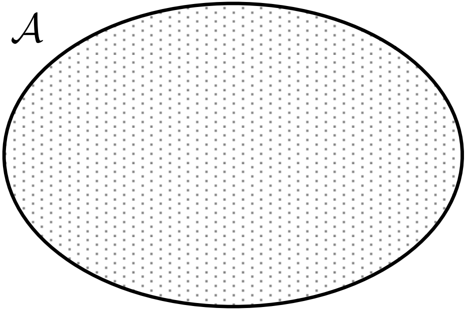
Epistemic Game Theory

- *Interactive Epistemology*: mathematical modelling of knowledge and belief of multiple interacting agents.
- *Epistemic Game theory*: interactive epistemology applied in the context of game theory.

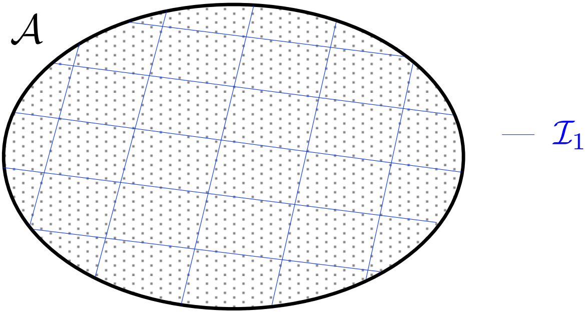
Epistemic Game Theory

- *Interactive Epistemology*: mathematical modelling of knowledge and belief of multiple interacting agents.
- *Epistemic Game theory*: interactive epistemology applied in the context of game theory.

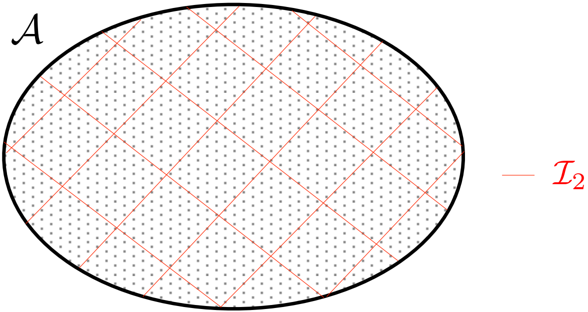
Aumann Structures: set-based approach



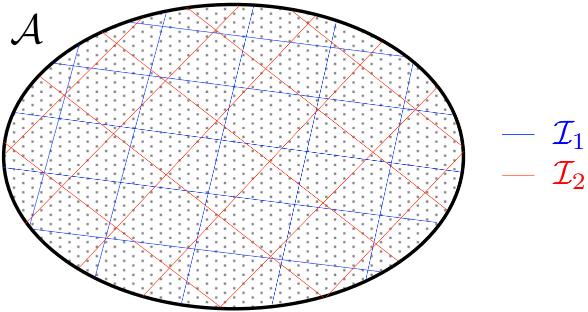
Aumann Structures: set-based approach



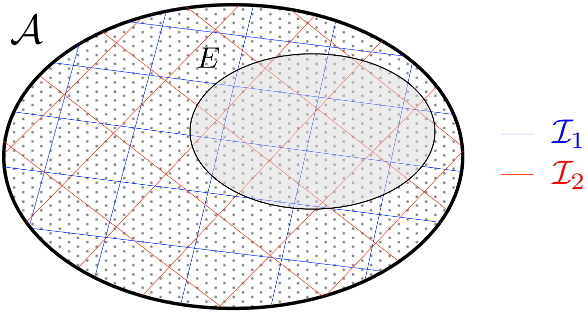
Aumann Structures: set-based approach



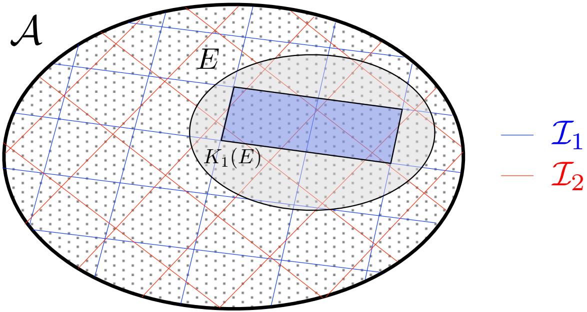
Aumann Structures: set-based approach



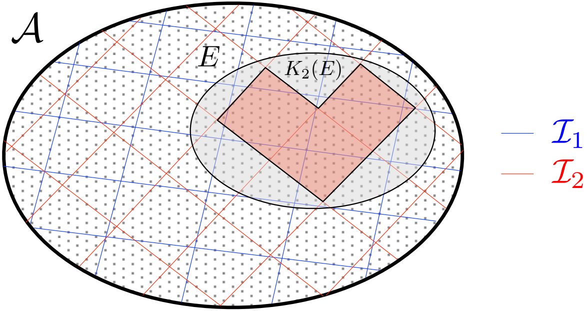
Aumann Structures: set-based approach



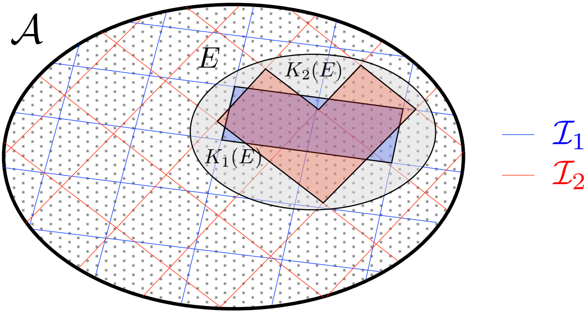
Aumann Structures: set-based approach



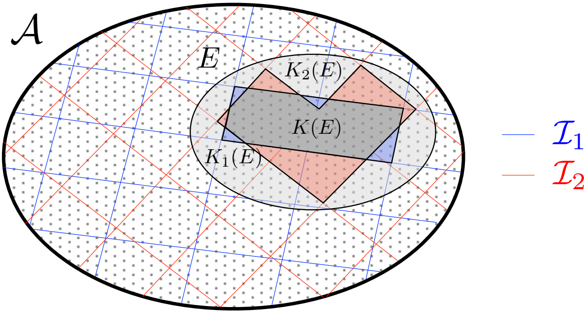
Aumann Structures: set-based approach



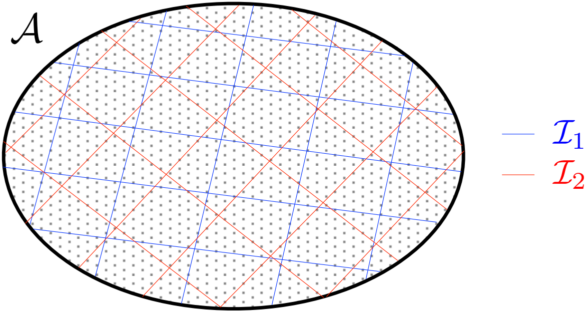
Aumann Structures: set-based approach



Aumann Structures: set-based approach



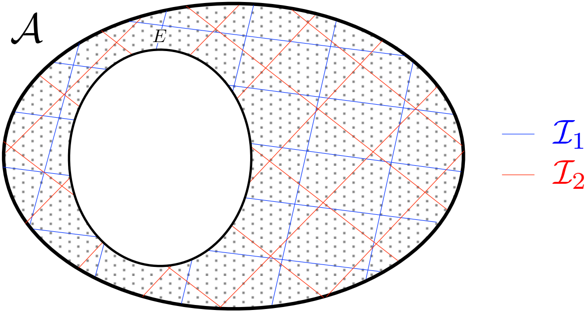
Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

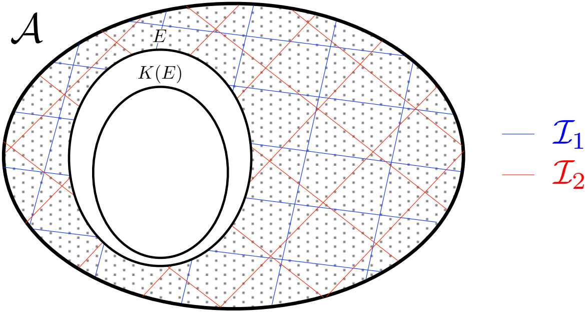
Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

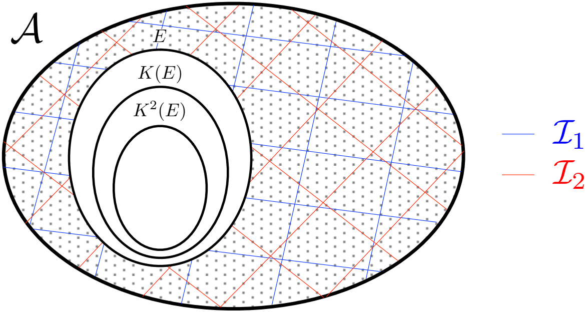
Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

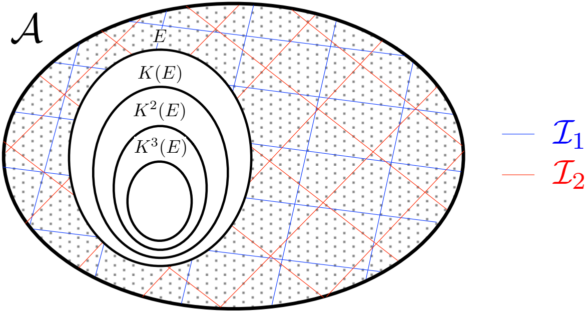
Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

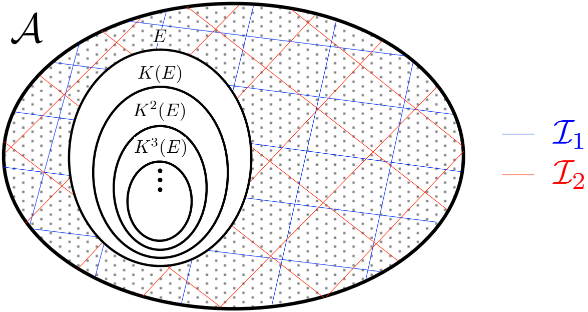
Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

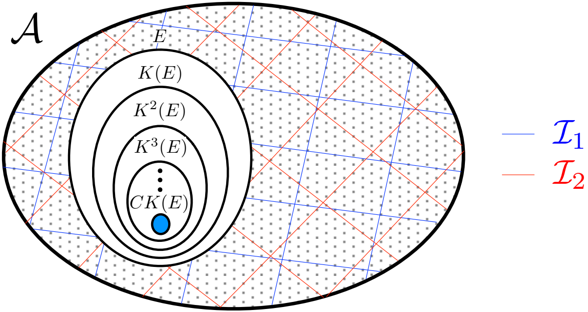
Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

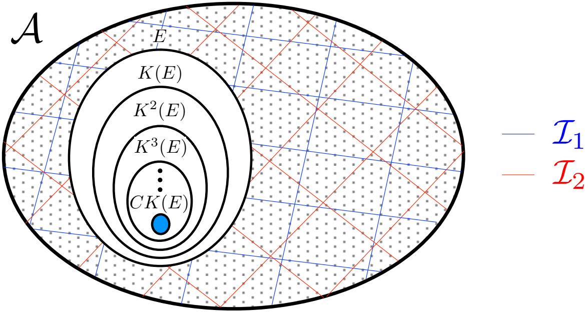
Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

Common Knowledge



$$CK(E) = \bigcap_{m \geq 0} K^m(E)$$

E is *common knowledge* iff everybody knows E, and everybody knows that everybody knows E, and everybody knows that everybody knows that everybody knows E, etc.

Limit Knowledge

- It might be thought of that the higher-order mutual knowledge claims $K^m(E)$ become closer and closer to common knowledge $CK(E)$. But it is not the case...
- In fact, the standard set-based approach to interactive epistemology lacks a general framework providing some formal notion of closeness between events.
- An additional topological dimension introduces a perception of closeness between events.

Limit Knowledge

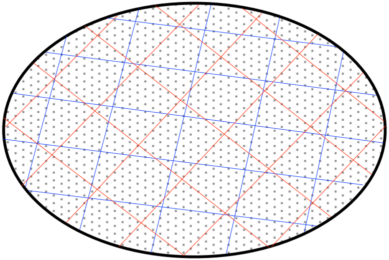
- It might be thought of that the higher-order mutual knowledge claims $K^m(E)$ become closer and closer to common knowledge $CK(E)$. But it is not the case...
- In fact, the standard set-based approach to interactive epistemology lacks a general framework providing some formal notion of closeness between events.
- An additional topological dimension introduces a perception of closeness between events.

Limit Knowledge

- It might be thought of that the higher-order mutual knowledge claims $K^m(E)$ become closer and closer to common knowledge $CK(E)$. But it is not the case...
- In fact, the standard set-based approach to interactive epistemology lacks a general framework providing some formal notion of closeness between events.
- An additional topological dimension introduces a perception of closeness between events.

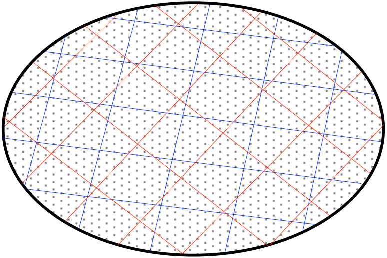
Limit Knowledge

$$\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in I}, p)$$

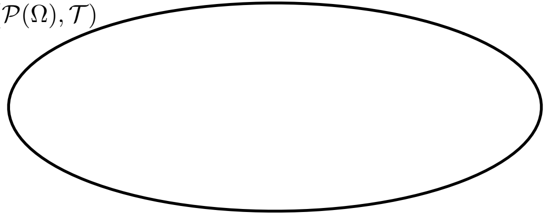


Limit Knowledge

$\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in I}, p)$

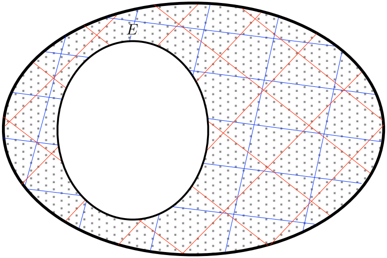


$(\mathcal{P}(\Omega), \mathcal{T})$

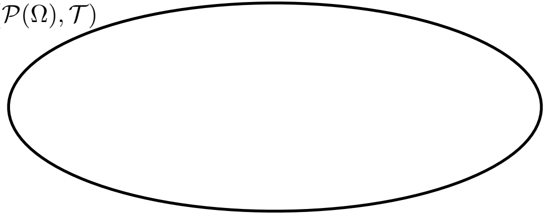


Limit Knowledge

$\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in I}, p)$

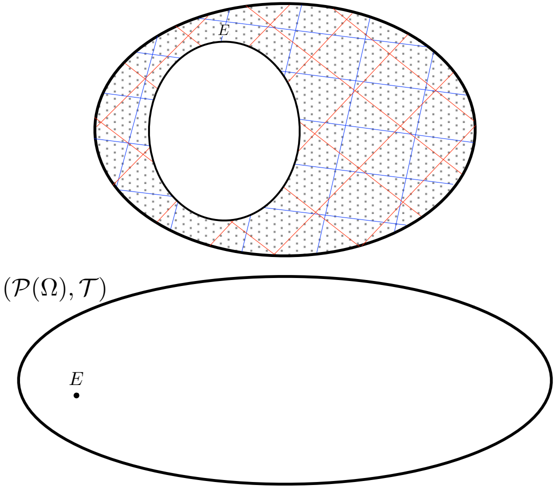


$(\mathcal{P}(\Omega), \mathcal{T})$

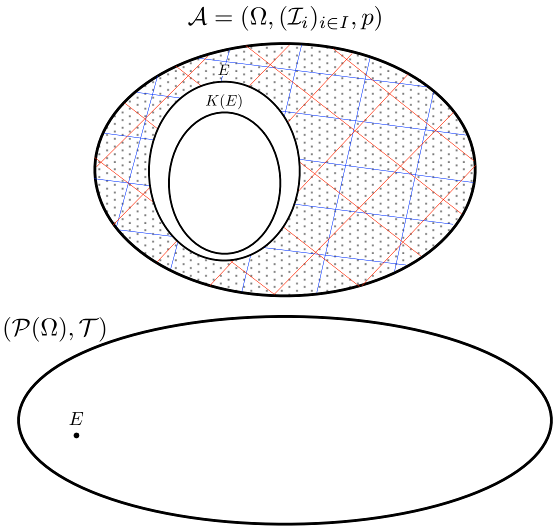


Limit Knowledge

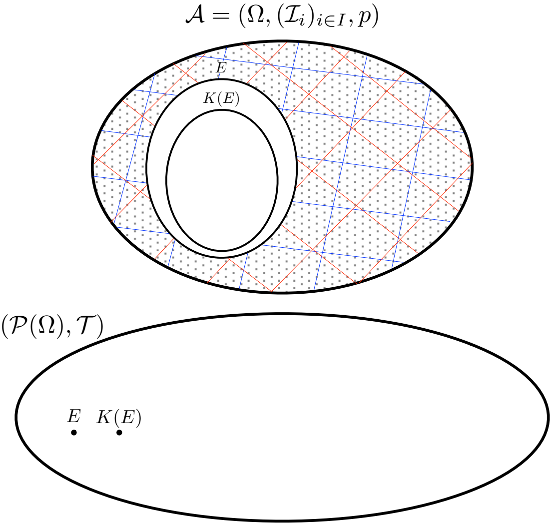
$$\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in I}, p)$$



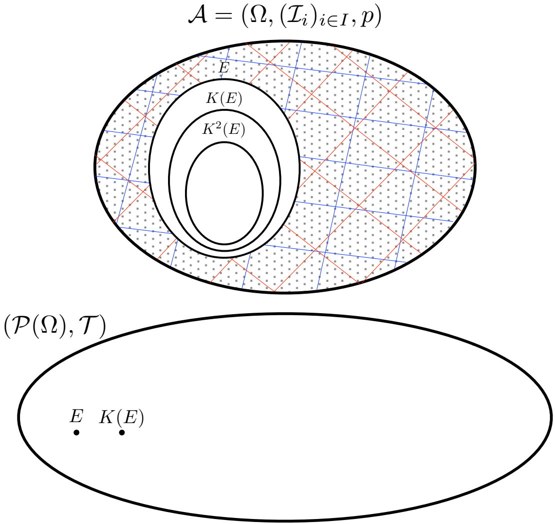
Limit Knowledge



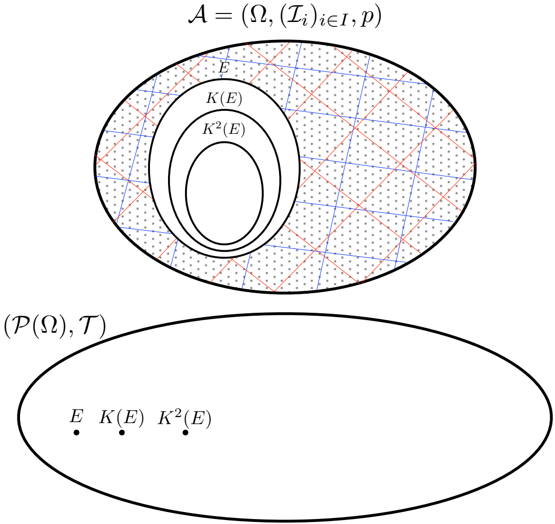
Limit Knowledge



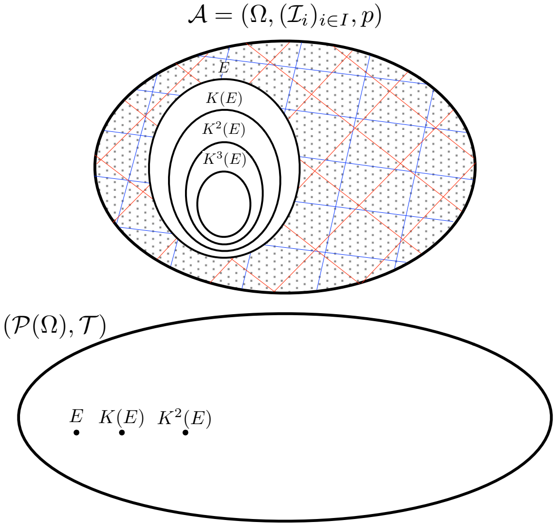
Limit Knowledge



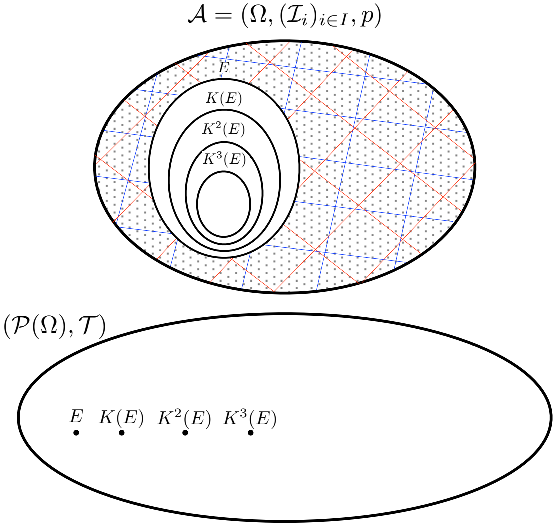
Limit Knowledge



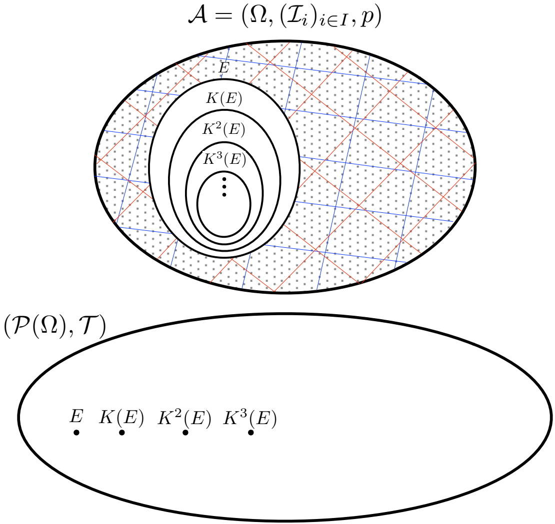
Limit Knowledge



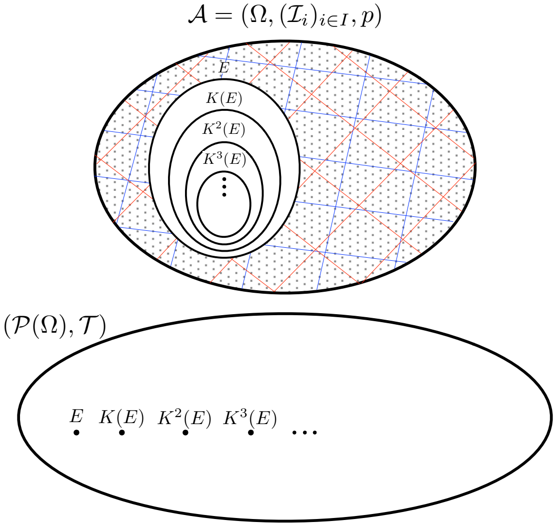
Limit Knowledge



Limit Knowledge

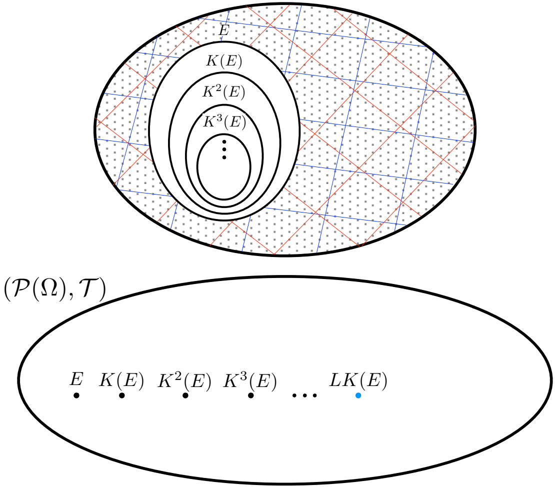


Limit Knowledge

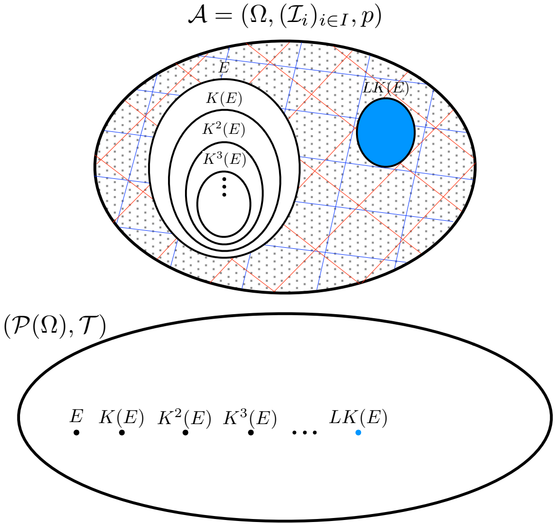


Limit Knowledge

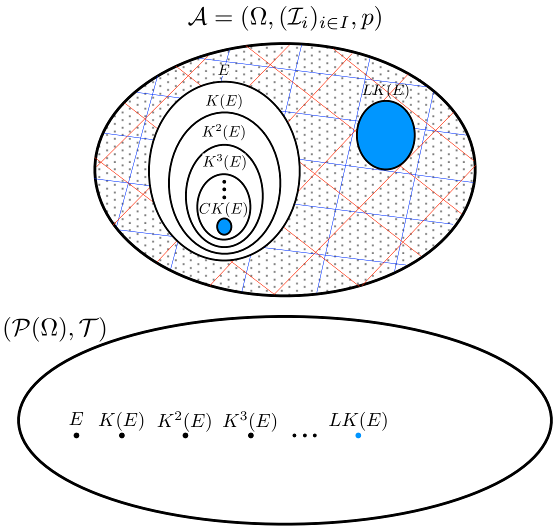
$\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in I}, p)$



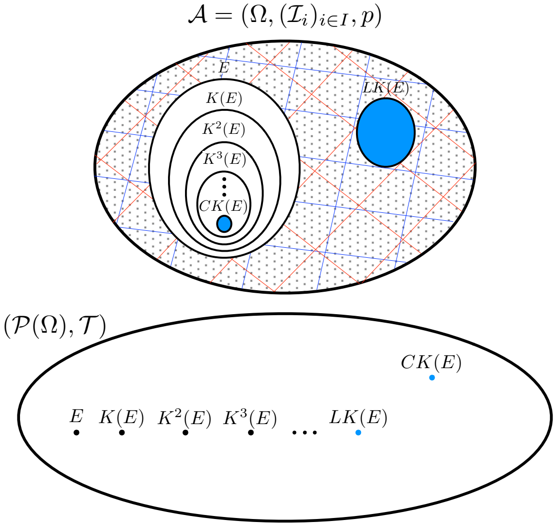
Limit Knowledge



Limit Knowledge



Limit Knowledge



Limit Knowledge

We now consider the following epistemic-topological operator *limit knowledge*.

Definition 1 (Limit Knowledge)

Let $\mathcal{A}$ be a topological Aumann structure, and E be some event. If the (topological) limit point of the sequence of iterated mutual knowledge claims $(K^m(E))_{m>0}$ is unique, then

$$LK(E) := \lim_{m \to \infty} K^m(E)$$

is the event that E is *limit knowledge* among the set I of agents.

Limit Knowledge

We now consider the following epistemic-topological operator *limit knowledge*.

Definition 1 (Limit Knowledge)

Let $\mathcal{A}$ be a topological Aumann structure, and E be some event. If the (topological) limit point of the sequence of iterated mutual knowledge claims $(K^m(E))_{m>0}$ is unique, then

$$LK(E) := \lim_{m\to\infty} K^m(E)$$

is the event that E is *limit knowledge* among the set I of agents.

Limit Knowledge

We now consider the following epistemic-topological operator *limit knowledge*.

Definition 1 (Limit Knowledge)

Let $\mathcal{A}$ be a topological Aumann structure, and E be some event. If the (topological) limit point of the sequence of iterated mutual knowledge claims $(K^m(E))_{m>0}$ is unique, then

$$LK(E) := \lim_{m \to \infty} K^m(E)$$

is the event that E is *limit knowledge* among the set I of agents.

Limit Knowledge

We now consider the following epistemic-topological operator *limit knowledge*.

Definition 1 (Limit Knowledge)

Let $\mathcal{A}$ be a topological Aumann structure, and E be some event. If the (topological) limit point of the sequence of iterated mutual knowledge claims $(K^m(E))_{m>0}$ is unique, then

$$LK(E) := \lim_{m \to \infty} K^m(E)$$

is the event that E is *limit knowledge* among the set I of agents.

Limit Knowledge

Limit knowledge can be understood as the event which is approached by the sequence of iterated mutual knowledge, according to some notion of closeness between events (provided by a topology on the event space).

Limit Knowledge, Games, and Aumann's Agreement Theorem

- *Limit knowledge* can provide alternative epistemic-topological characterizations of solution concepts in games.
- *Limit knowledge of rationality* can even potentially characterize any possible event and solution concept.
- Aumann's Agreement Theorem (1976): "Agents cannot agree to disagree". Limiting result: No-Trade Theorem. Study of weakened or modified epistemic assumptions:
- We proved that "agents can limit-agree to disagree".

Limit Knowledge, Games, and Aumann's Agreement Theorem

- *Limit knowledge* can provide alternative epistemic-topological characterizations of solution concepts in games.
- *Limit knowledge of rationality* can even potentially characterize any possible event and solution concept.
- Aumann's Agreement Theorem (1976): "Agents cannot agree to disagree". Limiting result: No-Trade Theorem: Milgrom and Stokey (1982); Study of weakened or modified epistemic assumptions: Geanakoplos and Polemarchakis (1982), Bacharach (1985), Monderer and Samet (1989), Samet (1990), Sonsino (1995), Neeman (1996a, 1996b), Kajii and Morris (1997), Morris (1999).
- We proved that "agents can limit-agree to disagree"...

Limit Knowledge, Games, and Aumann's Agreement Theorem

- *Limit knowledge* can provide alternative epistemic-topological characterizations of solution concepts in games.

- *Limit knowledge of rationality* can even potentially characterize any possible event and solution concept.

- Aumann's Agreement Theorem (1976): "Agents cannot agree to disagree". Limiting result: No-Trade Theorem: Milgrom and Stokey (1982); Study of weakened or modified epistemic assumptions: Geanakoplos and Polemarchakis (1982), Bacharach (1985), Monderer and Samet (1989), Samet (1990), Sonsino (1995), Neeman (1996a, 1996b), Kajii and Morris (1997), Morris (1999).

- We proved that "agents can limit agree to disagree".

Limit Knowledge, Games, and Aumann's Agreement Theorem

- *Limit knowledge* can provide alternative epistemic-topological characterizations of solution concepts in games.

- *Limit knowledge of rationality* can even potentially characterize any possible event and solution concept.

- Aumann's Agreement Theorem (1976): "Agents cannot agree to disagree". Limiting result: No-Trade Theorem: Milgrom and Stokey (1982); Study of weakened or modified epistemic assumptions: Geanakoplos and Polemarchakis (1982), Bacharach (1985), Monderer and Samet (1989), Samet (1990), Sonsino (1995), Neeman (1996a, 1996b), Kajii and Morris (1997), Morris (1999).

- We proved that "agents can limit-agree to disagree".

Limit Knowledge, Games, and Aumann's Agreement Theorem

- *Limit knowledge* can provide alternative epistemic-topological characterizations of solution concepts in games.
- *Limit knowledge of rationality* can even potentially characterize any possible event and solution concept.
- Aumann's Agreement Theorem (1976): "Agents cannot agree to disagree". Limiting result: No-Trade Theorem: Milgrom and Stokey (1982); Study of weakened or modified epistemic assumptions: Geanakoplos and Polemarchakis (1982), Bacharach (1985), Monderer and Samet (1989), Samet (1990), Sonsino (1995), Neeman (1996a, 1996b), Kajii and Morris (1997), Morris (1999).
- We proved that "agents can limit-agree to disagree".

Limit Knowledge, Games, and Aumann's Agreement Theorem

- *Limit knowledge* can provide alternative epistemic-topological characterizations of solution concepts in games.
- *Limit knowledge of rationality* can even potentially characterize any possible event and solution concept.
- Aumann's Agreement Theorem (1976): "Agents cannot agree to disagree". Limiting result: No-Trade Theorem: Milgrom and Stokey (1982); Study of weakened or modified epistemic assumptions: Geanakoplos and Polemarchakis (1982), Bacharach (1985), Monderer and Samet (1989), Samet (1990), Sonsino (1995), Neeman (1996a, 1996b), Kajii and Morris (1997), Morris (1999).
- We proved that "agents can limit-agree to disagree"...

Conclusion

- Limit knowledge is a new epistemic-topological operator capable of capturing reasoning patterns of agents based on closeness of events.
- The operator has relevant implications in interactional situations or games.
- Future work: topological foundation of common knowledge.

Conclusion

- Limit knowledge is a new epistemic-topological operator capable of capturing reasoning patterns of agents based on closeness of events.
- The operator has relevant implications in interactional situations or games.
- Future work: topological foundation of common knowledge.

Conclusion

- Limit knowledge is a new epistemic-topological operator capable of capturing reasoning patterns of agents based on closeness of events.
- The operator has relevant implications in interactional situations or games.
- Future work: topological foundation of common knowledge.