Habilitation à Diriger des Recherches
(HDR)

• • •

Part I
Computational Capabilities of Recurrent Neural Networks

•

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

JÉRÉMIE CABESSA

Université Panthéon-Assas – Paris II
Laboratoire d'Économie Mathématique
et de Microéconomie Appliquée (LEMMA)
75006 Paris, France

June 2016

ABSTRACT (PART I)

How does the brain encode and process information? Assuming that at least some aspects of the brain processes are of a computational nature, the following questions naturally arise. What are the computational capabilities of the neural networks involved? Do they transcend the limits of classical Turing machines? Is there an upper bound on this computational power?

Understanding the computational and dynamical capabilities of biological neural networks represents a most challenging issue, with considerable repercussions, ranging from theoretical and philosophical considerations to practical implications in the fields of artificial intelligence, machine learning, bio-inspired computing, robotics, etc. In this context, the theoretical computer scientist approach to neural computation has been focused on comparing the computational powers of diverse neural network models with those of abstract computing devices. Nowadays, the computational capabilities of neural models are known to be tightly related to the kind of activation functions used by the neurons, to the nature of their synaptic connections, to the eventual presence of noise in the model, to the possibility for the neural architecture to evolve over time, etc. They have been shown to range from the finite state automaton up to the super-Turing level.

In this manuscript, we review some important results concerning the computational capabilities of various models recurrent neural networks involved in different paradigms of computation: classical, interactive, and attractor-based. Overall, the Boolean neural networks are computationally equivalent to finite state automata. The static rational-weighted and real-weighted neural networks are equivalent to Turing machines and Turing machines with advices, respectively. The evolving neural networks are also equivalent to Turing machines with advices – and hence to static real-weighted networks – irrespective of whether their synaptic weights are modelled by rational or real numbers, and their patterns of evolvability restricted to binary updates or expressed by any other more general form of updating. Accordingly, analog and evolving recurrent neural networks are super-Turing.

These considerations show that recurrent neural networks represent a natural model of computation beyond the scope of classical Turing machines. The incorporation of either the power of the continuum or of some minimal evolving capabilities in a basic neural model provide alternative and equivalent ways towards the achievement of maximal computational potentialities. In fact, while Turing-equivalent neural models can only capture brain-like systems that are discrete, based on bit-calculations, and fixed in their architectures, super-Turing neural models can, by contrast, describe structures involving continuous levels of chemicals, comprizing adaptive and evolving architectures, and disclosing non-linear dynamical properties that are most relevant to brain dynamics, such as rich chaotic behaviors. Yet, at the current time, the critical issue of the possible achievement and exploitability of those hypercomputational capabilities by biological or artificial neural networks remains beyond reach.

RÉSUMÉ (PARTIE I)

Notre cerveau, cet organe essentiel qui assure la régulation de toutes nos fonctions vitales, n'a de cesse de traiter en parallèle une multitude de flux informationnels lui provenant de tout le corps humain. Mais comment procède-t-il ? Comment traite-t-il cette information ? Quels sont les processus mis en oeuvre dans le codage, le décodage et la transmission de cette information ? En supposant que certains aspects de ce traitement informationnel soient de nature computationnelle, les questions suivantes se présentent alors. Quels seraient les capacités calculatoires des réseaux de neurones impliqués dans ces processus ? Sommes nous en présence, et donc porteur, d'un pouvoir computationnel qui se situerait au-delà de celui des machines de Turing ? Existe-t-il une borne supérieure à ce pouvoir computationnel ?

La compréhension fondamentale des processus computationnels neuronaux représente un enjeu majeur de la recherche scientifique actuelle, avec d'importantes répercussions non seulement théoriques, dans le cas de considérations philosophico-scientifiques ayant trait au concept de computation ou à l'intelligence biologique en général, mais également pratiques, de par ses répercussions dans le vaste domaine de l'intelligence artificielle, et en particulier, du « machine learning ».

Dans ce contexte, des travaux d'importance majeure ont montré que divers modèles de réseaux neuronaux possèdaient des capacités computationnelles allant du niveau des automates finis, à celui des machines de Turing, jusqu'à celui de certaines machines de Turing avec oracles particuliers. Ainsi, les réseaux de neurones constituent un modèle de calcul dont les potentialités transcendent celles des machines de Turing, pour se situer au niveau hypercomputationnel, ou super-Turing. Plus généralement, il a également été montré que divers systèmes dynamiques, qu'ils soient analogiques, quantiques, ou gouvernés par certaines lois de la physique classique ou relativistes pouvaient également exhiber des comportements qui soient non Turing-calculables.

Dans ce manuscrit, je propose de présenter quelques résultats théoriques majeurs concernant le pouvoir computationnel des réseaux de neurones récurrents. Ces résultats s'étalent sur une période de soixante-dix ans, du début des années quarante jusqu'à la période actuelle.

Les chapitres 2, 3 et 4 fournissent respectivement les pré-requis mathématiques, les définitions des modèles de calculs abstraits, ainsi que la présentation générale du concept de réseaux de neurones nécessaires à la formalisation des résultats présentés dans ce travail.

Le chapitre 5 concerne l'étude des capacités computationnelles des réseaux de neurones impliqués dans un cadre de computation dite « classique », c'est-à-dire lorsque les systèmes sont vus comme des « boîtes noires fonctionnelles » qui transforment des entrées finies en sorties finies. Dans ce contexte, McCulloch et Pitts, Kleene et Minsky ont montré que les réseaux de neurones Booléens sont computationnellement équivalents aux automates finis. Siegelmann et Sontag ont, quant à eux, montré que lorsque les fonctions d'activation des neurones étaient étendues du cas Booléen à un contexte sigmoïde, les réseaux correspondants voyaient leur capacités calculatoires s'accroître de manière drastique. Plus précisément, les réseaux sigmoïdaux à poids rationnels se trouvent être équivalents à des machines de Tu-

ring, et ceux à poids réels – appelés réseaux de neurones analogiques – témoignent de potentialités équivalentes à celles des machines de Turing avec conseils (un cas particulier de machines de Turing avec oracle). En ce sens précis, les réseaux de neurones analogiques représente un modèle de calcul qualifié de super-Turing.

Dans cette perspective, nous avons montré que les réseaux de neurones dits « évolutifs » – i.e., pourvus d'une certaine capacité d'évolution de leur poids synaptiques, ou, plus généralement, de leur architecture globale – étaient également computionnellement équivalents à des réseaux statiques à poids réels, et donc, à des machines de Turing avec conseils. De plus, cette équivalence reste valable d'une part, indépendamment du fait que les poids synaptiques évolutifs soient contraints à des patterns d'évolutions simplistes, à savoir binaires, ou au contraire, totalement généraux, sans restriction aucune ; et, d'autre part, indépendamment du fait que les poids aussi bien statiques qu'évolutifs soient modélisés par des nombres rationnels ou réels. En résumé, les réseaux neuronaux statiques analogiques et ceux dits évolutifs – à poids rationnels ou réels et à évolution binaire ou générale – sont tous super-Turing équivalents.

Le chapitre 6 généralise ces notions dans le cadre de la computation dite interactive, où, plutôt que de se comporter de manière purement fonctionnelles, les systèmes procèdent à un échange séquentiel et continu d'information avec leur environnement. Les réseaux sont alors considérés comme des ω-transducteurs qui traduisent pas à pas des suites de bits infinies en suites de bits finies ou infinies. Dans ce contexte, les réseaux statiques rationnels sont équivalents à des machines de Turing interactives et réalisent les ω-translations dites récursives continues. Les réseaux statiques réels ainsi que ceux évolutifs possèdent les mêmes capacités calculatoires que celles des machines de Turing avec conseils ; ils réalisent la classe des ω-translations continues.

Le chapitre 7 traite, quant à lui, d'un nouveau type de computation neuronale basée sur le concept d'attracteur. Dans ce cadre, il est supposé que les réseaux neuronaux sont munies d'un certain nombre de cellules Booléennes spécifiques qui, lorsque le réseau est soumis à flot d'inputs infini, entreront forcément dans une dynamique d'attracteur. Les attracteurs sont alors supposés être catégorisés en deux types distincts : acceptant ou rejetant. L'ω-langage d'un réseau neuronal est alors défini comme l'ensemble des flots d'inputs infinis qui induisent un attracteur acceptant, et le pouvoir computationnel des réseaux est caractérisé par la complexité topologique de leurs ω-langages sous-jacents.

Selon cette approche, les réseaux Booléens sont équivalents à des automates de Büchi ou de Muller. Par conséquent, la plus fine des hiérarchisations d'ω-langages peut se transposer du contexte de la théorie des automates à celui des réseaux de neurones, induisant une nouvelle notion de complexité basée sur la dynamique attractive des réseaux. Une illustration de cette notion de complexité est fournie dans le cadre de l'étude d'un modèle Booléen d'un réseau cérébral important.

Dans le cas sigmoïdal, les réseaux déterministes à poids rationnels sont équivalents à des machines de Turing de Muller déterministes. Les réseaux déterministes à poids réels ou évolutifs reconnaissent, quant à eux, la classe de toutes les combinaisons Booléennes des ω-langages $\mathbf{\Pi_2^0}$ ($BC(\mathbf{\Pi_2^0})$), et, à ce titre, possèdent des capacités super-Turing.

Par ailleurs, deux types de non-déterminisme sont considérés. Dans le premier cas, les réseaux sont munis d'une cellule Booléenne, une « guess cell », qui définira la condition de non-déterminisme en question. Dans le deuxième cas, les réseaux sont pourvus d'un « espace d'évolution » contenant en puissance tous les patterns d'évolution possibles de leurs poids synaptiques. Au début de chaque computation, le réseau sélectionne au sein de cet espace, de manière non-déterministe, une évolution spécifique pour ses poids synaptiques, à laquelle il se tiendra ensuite pour toute la durée infinie de la computation à venir. Six modèles de réseaux non-déterministes de type I et quatre modèles de type II sont alors considérés en fonction de la nature rationnelle/réelle et statique/évolutive de leurs poids synaptiques. Dans ce contexte, les réseaux statiques rationnels de type I sont computationellement équivalents à des machines de Turing de Muller non-déterministes, et reconnaissent donc la classe des ω-langages analytiques effectifs (Σ_1^1 lightface). Les neuf autres modèles de types I et II sont équivalents entre eux et reconnaissent la classe de tous les ω-langages analytiques ($\mathbf{\Sigma_1^1}$ boldface). Ils sont donc super-Turing également.

Ces résultats témoignent du fait que les réseaux de neurones récurrents constituent un modèle de computation naturel au-delà de la barrière du Turing-calculable. De telles considérations sont parfois évoquées, de manière controversée, pour justifier une certaine supériorité de l'intelligence biologique par rapport à l'intelligence artificielle. Néanmoins, il paraît raisonnable d'oser affirmer que le modèle super-Turing des machines de Turing avec conseils semble particulièrement approprié à capturer certains aspects intrinsèques à la computation neuronale. En effet, alors que les modèles Turing équivalents ne décrivent des systèmes dynamiques qui ne sont que discrets et d'architectures statiques, le modèle super-Turing peut, en revanche, intégrer la modélisation de variables continues, d'architectures adaptatives ou évolutives, ainsi que de propriétés biologiques cruciales, tels que certains aspects de non-linéarité et de chaoticité.

Les capacités hypercomputationnelles des réseaux évolutifs, bien qu'équivalentes à celles des réseaux analogiques, présentent un intérêt particulier. En effet, alors que le pouvoir du continu apparaît comme un concept mathématique au service de la modélisation de phénomènes biologiques ou physiques, les capacités d'évolution et de plasticité constituent, à l'inverse, des phénomènes réellement observables. Toutefois, l'existence d'une telle hypercomputationalité dépend de possibilité qu'aurait la « nature » à réaliser des patterns d'évolution non-récursifs, sans quoi les réseaux ne seraient réduits qu'à des systèmes qui soient Turing-équivalents.

La supposition que la nature n'impliquerait pas seulement des patterns préprogrammés, mais également certains phénomènes non-récursifs, tels que la stochasticité pure par exemple, permet d'affirmer l'existence de capacités hypercomputationnelles ou super-Turing. Mais à l'heure actuelle, les questionnements philosophiques et scientifiques à propos tant de la possibilité d'existence que de l'exploitabilité de telles capacités computationnelles demeurent à leur balbutiement.

ABSTRACT (PART II)

Nowadays, *game theory* has become a major field of research, mainly used in mathematical economics – towards the modelling of competing behaviors of interacting agents –, but also with considerable applications in logic, computer science, biology, political science, and psychology. In this general framework, formal *interactive epistemology* provides a general framework in which epistemic notions – such as knowledge, belief and subsequent concepts – can be modelled for situations involving multiple agents. When employed in the specific context of game-playing agents, the discipline, referred to as *epistemic game theory*, studies the behavioral implications of such epistemic hypotheses in games. Here, we follow Aumann's set-based approach to epistemic game theory, which formalizes epistemic notions via the consideration of set-theoretic tools.

In this context, the epistemic operator of "common knowledge" has proven to be of a specific relevance, in being considered either as a basic or as an epistemic hypothesis in various interactive or game-theoretic situations. But important results have also unveiled the limitations and paradoxes linked to this operator, and led to the study of alternative notions of shared knowledge.

Along this line, we introduced the novel epistemic-topological operator *limit knowledge*, defined as the topological limit of all higher-order mutual knowledge claims, and accordingly, linked to both the epistemic as well as the topological features of the underlying semantics. We showed that our epistemic-topological operator "limit knowledge" does genuinely differ from "common knowledge", and also from "almost common knowledge" and "common p-belief".

Subsequently, we proved that the operator "limit knowledge" is actually capable of relevant characterizations of solution concepts in games. In fact, we provided an example of a concrete Cournot-type game where limit knowledge of rationality strictly refines common knowledge of rationality, in terms of solution concepts. Morevoer, we generally showed that, for any given game and epistemic model of it satisfying some appropriate epistemic-topological conditions, limit knowledge of rationality is actually capable of characterizing any possible solution concept.

Besides, we revisited Aumann's seminal "agreement theorem" in light of a novel epistemic-topological hypothesis induced by limit knowledge. We showed that, when the hypothesis of common knowledge of their posteriors is replaced by that of limit knowledge of their posteriors, the impossibility for the agents to agree to disagree does actually no longer hold.

These considerations argue in favor of a general topological approach to set-based interactive epistemology and epistemic game theory. It can be used to model some additional agents' perceptions of closeness between elements of the semantic structures, like worlds or events. In turn, it enables to capture broader reasoning patterns of interacting or game-theoretic agents, which not only depend on the epistemic but also on the topological features of the interactive situation.

RÉSUMÉ (PARTIE II)

L'épistémologie interactive fournit un cadre formel permettant de modéliser les concepts de connaissance et de croyance des agents impliqués dans diverses situations interactionnelles. Appliquée au cas particulier de la théorie des jeux, cette approche épistémique permet de modéliser des comportements stratégiques complexes dans lesquels divers aspects de connaissance et de croyance des agents peuvent être pris en compte. L'étude des fondements épistémiques de la théorie des jeux consiste alors à comprendre les relations qui existent entre, d'une part, les hypothèses épistémiques liées à certaines situations interactionnelles, et, d'autre part, les concepts de solutions qu'elles induisent en théorie des jeux. Dans ce contexte, l'approche ensembliste de l'épistémologie interactive et de la théorie des jeux épistémique, introduite et considérablement développée par Aumann (1976), (1987), (1995), (1996), (1998a,b), (1999a,b), (2005), fournit un formalisme particulièrement adéquat pour capturer les aspects de connaissance et de croyance d'agents impliqués dans diverses situations interactionnelles et stratégiques.

Dans ce contexte, l'opérateur épistémique de « connaissance commune » (« common knowledge ») introduit par Lewis (1969) possède une importance toute particulière, d'une part en tant qu'hypothèse de base – par exemple lorsque l'information des agents et la forme du jeu considéré est supposée être de connaissance commune parmi les agents en question (voir Aumann (1976) et (1999a)) –, et, d'autre part en tant qu'hypothèse épistémique – par exemple lorsqu'une certaine forme de rationalité est supposée être de connaissance commune parmi les agents (voir par exemple Bernheim (1984), Pearce (1984), TanWerlang (1988), Borgers (1993)). Intuitivement, un évènement E est dit de connaissance commune si tout le monde sait E et tout le monde sait que tout le monde sait E et tout le monde sait que tout le monde sait que tout le monde sait E, etc. ad infinitum. Autrement dit, le concept de connaissance commune d'un évènement E apparaît comme la conjonction (ou l'intersection) infinie des connaissance mutuelles d'ordres supérieurs de cet évènement.

D'importants résultats ont mis à jours les limitations et paradoxes du concept de connaissance commune (Aumann (1976), Milgrom et Stokey (1982), Morris (2002)), et, par conséquent, ont amené à étudier la pertinence de cette notion ainsi que la possibilité de considérer d'autre concepts de connaissances qui seraient plus adéquats selon les situations épistémiques considérées (voir par exemple Monderer et Samet (1989), Rubinstein (1989), Börgers (1994), Lipman(1994), Sonsino (1995), Neeman (1996a,b), Kajii et Morris (1997), Morris (1999)).

Mon travail de recherche en collaboration avec Christian W. Bach, présenté dans ce manuscrit, s'inscrit dans précisément dans cette perspective de reconsidération et d'approfondissement de l'opérateur épistémique de connaissance commune. Plus précisément, nous avons d'abord introduit le concept de « structures de Aumann topologiques » comme des structures de Aumann pourvues d'une topologie[1] additionnelle sur l'espace des évènements. De telles structures permettent alors de modéliser une certaine notion de « rapprochement » ou de « proximité »

[1] Un espace topologique représente une généralisation du concept d'espace métrique, et donc, permet de modéliser une généralisation du concept de distance entre ses éléments.

entre les évènements (il semble en effet naturel de considérer que, pour certains agents, l'évènement « il pleut » soit plus proche de l'évènement « il fait froid » que de l'évènement « il fait chaud », et ce bien qu'aucune relation de causalité ou d'implication stricte n'existe entre ces évènements). Nous avons ensuite introduit le nouvel opérateur épistémique de « connaissance limite » (« limit knowledge ») défini comme la limite topologique de la suite des opérateurs de connaissances mutuelles d'ordres supérieurs.

Nous avons alors pu montrer que dans le cas d'une structures de Aumann infinie, ce concept de connaissance limite diffère clairement de ceux de connaissance commune (common knowledge), de connaissance presque-commune (almost common knowledge) de Rubinstein (1989) et de p-croyance commune (common p-belief) de Monderer et Samet (1989).

Nous avons par la suite prouvé que l'opérateur épistémique de connaissance limite était capable de caractériser de manière pertinente de nouveaux concepts de solutions en théorie des jeux (voir Bach et Cabessa (2009) et (2011)). Plus précisément, nous avons construit un jeu et un modèle épistémique de celui-ci pour lesquels l'hypothèse épistémique de connaissance limite de la rationalité impliquait le concept de solution « iterated strict dominance » suivi d'une dernière itération de « weak dominance ». Nous avons également prouvé que sous certaines conditions de régularité, tout concept de solution peut, d'une manière ou d'une autre, être caractérisé par un certain type de connaissance limite de la rationalité des agents. Ces résultats témoignent d'une puissance de caractérisation de l'opérateur épsitémique de connaissance limite par rapport à celle de l'opérateur de connaissance commune.

D'autre part, nous avons montré que l'opérateur épistémique de connaissance limite était capable de revisiter le fameux théorème de Aumann (« The Agreement Theorem ») de manière pertinente (voir Bach et Cabessa (2012) et (2016)). En effet, dans le cas où des agents partageraient une connaissance limite plutôt que commune de leurs croyances postérieures, il leur deviendrait alors possible d'entretenir des croyances postérieures qui diffèrent. Ce résultat témoigne d'une certaine pertinence du concept de connaissance limite par rapport à celui de connaissance commune dans d'importantes situations épistémiques.

Le cadre épistémico-topologique formel développé dans notre étude apporte une dimension de « rapprochement » ou de « proximité » entre les évènements et permet, d'une part, de modéliser les concepts de connaissance et de croyance des agents de manière plus raffinée, et, d'autre part, de capturer des patterns de raisonnement des agents qui vont au-delà de considérations purement logiques.

Contents

List of Figures

Part I

Computational Capabilities of Recurrent Neural Networks

1 INTRODUCTION

How does the brain encode and process information? According to the computational theory of mind, the mind itself is a computational system. Leaving aside the debate between computationalism and connectionism, we assume that at least some aspects of the brain processes are of a computational nature. Therefore, the following questions naturally arise. What are the computational capabilities of the neural networks involved? Do they transcend the limits of classical Turing machines? Is there an upper bound on this computational power?

Understanding the computational and dynamical capabilities of biological neural networks is a most challenging issue, with considerable repercussions, ranging from theoretical and philosophical considerations to practical implications in the fields of artificial intelligence, machine learning, bio-inspired computing, general robotics, humanoid robotics, etc.

In this context, the theoretical computer scientist approach to neural computation has mainly been focused on comparing the computational powers of diverse theoretical neural models with those of abstract computing devices. Nowadays, the computational capabilities of neural models is known to be tightly related to the kind of the activation function used by the neurons, to the nature of their synaptic connections, to the eventual presence of noise in the model, and to the possibility for the neural architecture to evolve over time.

More precisely, these studies were initiated by McCulloch and Pitts in 1943 [118]. In their seminal work, they proposed a modeling of the nervous system as a finite interconnection of logical devices. For the first time, neural networks were considered as discrete abstract machines, and the issue of their computational capabilities investigated from the automata-theoretic perspective.

Along these lines, Kleene and Minsky proved that recurrent neural networks with Boolean activation functions are computationally equivalent to finite state automata [89, 119]. In Minsky's own words [119]:

> It is evident that each neural network of the kind we have been considering is a finite-state machine. [...] It is interesting and even surprising that there is a converse to this. Every finite-state machine is equivalent to, and can be simulated by, some neural net.

This result turned out to be of specific relevance for the implementation of finite state machines on parallel hardwares, as well as for the incorporation of prior symbolic knowledge in neural networks, yielding to better learning performances. It opened the way to some important investigations concerning the simulation of fi-

nite state machines by various models of neural networks: first-order, second-order, feedforward, recurrent, with hard-threshold, sigmoid or radial activation functions, and with several other restrictions on their weights or architectures; and concerning the learning performances of these neural networks; see for instance [10, 11, 44, 50, 53, 54, 59, 66, 73, 92, 124, 125, 136, 152, 183, 189].

Besides, already in 1948, Turing made a significant step forward by showing the possibility of surpassing the capabilities of finite state machines – and reaching Turing universality – via some kinds of neural networks called *B-type unorganized machines*, consisting of specific interconnections of NAND neuronal-like units [169]. He suggested that the consideration of sufficiently large B-type unorganized machines could simulate the behavior of a universal Turing machine, up to some memory bound [169]:

> In particular with a B-type unorganized machine with sufficient units one can find initial conditions which will make it into a universal machine with a given storage capacity.

The Turing universality of neural networks involving infinitely many binary neurons has further been investigated in many directions, as for instance in [52, 56, 67, 135, 144]. Furthermore, Turing brilliantly anticipated the two concepts of "learning" and "training" that would later become central in machine learning [169].

> If we are trying to produce an intelligent machine, and are following the human model as closely as we can, we should begin with a machine with very little capacity to carry out elaborate operations or to react in a disciplined manner to orders (taking the form of interference). Then by applying appropriate interference, mimicking education, we should hope to modify the machine until he could be relied on to produce definite reactions to certain commands. [...] The process of setting up these initial conditions so that the machine will carry out some particular useful task may be called 'organizing' the machine. 'Organizing' it thus a form of 'modification'. [...] All of this suggests that the cortex of the infant is an unorganized machine, which can be organized by suitable interfering training, or something like it.

Ten years later, in the late 50's, von Neumann proposed a particularly relevant approach to the issue of information processing in the brain from the hybrid perspective of *digital* and *analog* computations [123]. He considered that the non-linear character of the operations performed by the brain emerges from a combination of discrete and continuous mechanisms. Accordingly, he envisioned neural computation as something strictly more powerful than abstract machines.

Almost at the same time, Rosenblatt proposed the perceptron as a more general computational neural model than the McCulloch-Pitts units [140]. The significant innovation consisted in the introduction of numerical synaptic weights as well as of a special pattern of interconnection. This neural model gave rise to the fundamental and still valid algorithmic conception of *learning* – which is achieved by adjusting the synaptic weights of the neuronal connections according to some specific task to be completed. The first learning algorithms, proposed by Rosen-

blatt [141] and Widrow [187], were concerned with two-layer feedforward networks. But Minsky and Papert analyzed the limited capabilities of the perceptron and notably showed that single-layer perceptrons were unable to implement the logical XOR [120]. These results dampen the enthusiasm of several researchers in the field.

These drawbacks lasted until Kohonen suggested to take into account Hebb's rule [68] in learning algorithms. He introduced a system model composed of at least two interacting subsystems of different nature, a competitive neural net and another subsystem modifying the local synaptic plasticity of the neurons in learning, leading to the implementation an effective and robust self-organizing system [90, 91]. Fukushima introduced the "neocognitron", a hierarchical multilayered neural network capable of robust visual pattern recognition through learning [55]. And notably, Rumerlhart et al. proposed the famous "backpropagation" algorithm as a novel learning technique: the synaptic weights of the network are continuous variables, updated by backward induction according to some gradient descent process [142]. Deep learning methods for neural networks is nowadays a tremendously active field of research; for a very detailed overview, see [143] and the references therein. Besides this, from a theoretical perspective, feedforward neural networks turn out to have universal approximation capabilities: they can approximate any well-behaved continuous or Borel real function to any degree of accuracy [74, 75].

In 1994 and 1995, Siegelmann and Sontag achieved two significant breakthroughs in the field. First, based on biological considerations, they chose to focus on sigmoidal rather than Boolean neurons, and proved the possibility of reaching Turing universality with finite recurrent neural networks of that kind. In fact, by considering rational synaptic weights and by extending the activation functions of the cells from Boolean to linear-sigmoid, the neural networks have their computational power drastically increased from the finite state automaton up to the Turing machine level [76, 122, 155]. Kilian and Siegelmann generalized this Turing universality to a broader class of sigmoidal neural networks [88]. The computational equivalence between so-called *rational recurrent neural networks* and Turing machines has now become a standard result in the field.

Secondly, following von Neumann considerations, they assumed that the variables appearing in the underlying chemical and physical phenomena could be modeled by continuous rather than discrete (rational) numbers. Accordingly, they proposed a precise study of the computational power of recurrent neural networks from the perspective of analog computation [151]. They introduced the concept of an *analog recurrent neural network* as a classical linear-sigmoid neural net equipped with real-weighted instead of rational-weighted synaptic connections. They proved that such analog recurrent neural networks are computationally equivalent to Turing machines with advices, and hence, capable of super-Turing computational power from polynomial time of computation already [150, 154]. This analog information processing model turns out to be capable of capturing non-linear dynamical properties that are most relevant to brain dynamics, such as rich chaotic behaviors [79, 84, 166, 167, 168], as well as dynamical and idealized chaotic systems that cannot be described by the universal Turing machine model [149]. A transfinite classification of analog recurrent neural networks based on the Kolmogorov complexity of

their underlying real synaptic weights has further been described [17]. According to these considerations, Siegelmann and Sontag formulated the so-called *Thesis of Analog Computation* – an analogous to the Church-Turing thesis, but in the realm of analog computation – stating that no reasonable abstract analog device can be more powerful than first-order analog recurrent neural networks [149, 154]. These results are sometimes invoked to support the debatable claim that some intrinsic dynamical and computational features of neurobiological systems might be beyond the scope of standard artificial models of computation.

In the late 90's, the issue of the robustness of the computational power of neural networks subjected to various kinds of noise was addressed. It was shown that the presence of analog noise would generally strongly reduce the computational power of the underlying systems to that of finite state automata, or even below [18, 111, 115]. On the other hand, the incorporation of some discrete source of stochasticity would rather tend to increase or maintain the capabilities of the neural systems [153].

The next major contribution to the field was provided by Maass, who intensively studied the computational power of *networks of spiking neurons*. In this case, the computational states are encoded in the temporal differences between the neurons' spikes rather than in the activation values of the cells. Maass proved that single spiking neurons are strictly more powerful than single threshold gates [113, 114]. He further extensively studied the lower and upper bounds on the computational complexity of networks of classical and noisy spiking neurons, see [103, 104, 106, 108, 110, 112] and [107, 109], respectively. In addition, he showed that networks of spiking neurons turn out to be capable of simulating analog recurrent neural networks [105].

Besides, in 2000, Păun introduced the novel concept of a *P system* – a highly parallel abstract model of computation inspired from the membrane-like structure of the biological cell [129, 130]. This foundational work led to the emergence of a tremendously active field of research, involving from highly theoretical to very practical results, with countless publications, events, and research groups worldwide, see [163]. In this context, the computational capabilities of various models of so-called *neural P systems* were studied, see for instance [82, 128, 131, 132, 133]. In particular, neural P systems provided with a bio-inspired source of acceleration were shown to be capable of hypercomputational capabilities, by spanning all levels of the arithmetical hierarchy [37, 58].

For more references and details about the computational power and complexity of neural networks, one could refer to the deep survey by Šìma and Orponen [156].

But the neural models considered up to that point were generally oversimplified, lacking many biological features which may be essential to the processing of information in the real brain.

For instance, as a first point, it is nowadays widely admitted that biological mechanisms like synaptic plasticity, cell birth and death, and changes in connectivity are intimately related to the storage and encoding of memory traces in the central nervous system, and provide the basis for most models of learning and memory in neural networks [1, 38, 42, 117, 138]. In the context of AI, the consideration of such evolving neural architectures in so-called *Evolving Connectionist Systems*

(ECoS) has proven to be fruitful, and significantly increased in applications in the recent years [85, 184].

Following these considerations, we studied the computational capabilities of a more biologically oriented recurrent neural model where the synaptic weights, the connectivity pattern, and the number of neurons can evolve rather than stay static. We showed that the so-called *evolving recurrent neural networks* are computationally equivalent to Turing machines with advices [24, 26]. This equivalence holds irrespective of whether the synaptic weights of the networks are modelled by rational or real numbers, and the patterns of evolvability[1] restricted to binary updates or expressed by any other more general form of updating. Consequently, the evolving and the analog recurrent neural networks disclose the same super-Turing computational power. It follows that the incorporation of either minimal evolving capabilities or of the power of the continuum in a basic neural model provide alternative and equivalent ways towards the achievement of maximal computational potentialities. This feature is of specific interest, since evolving capabilities of neural networks are observable in nature, as opposed to the power of the continuum which remains at a conceptual level.

Secondly, it has been argued that the classical computational approach [171] "no longer fully corresponds to the current notion of computing in modern systems" [97] – especially when it refers to bio-inspired complex information processing systems. In the brain (or in organic life in general), information is rather processed in an interactive way [61, 185]: previous experience must affect the perception of future inputs, and older memories may themselves change with response to new inputs.

Accordingly, we studied the computational capabilities of recurrent neural networks involved in an interactive paradigm of computation, where information is processed via a sequential exchange of information between the system and its environment [95, 98]. We showed that similarly to the classical computational context, the rational-weighted *interactive recurrent neural networks* are computationally equivalent to interactive Turing machines; the real-weighted and the evolving *interactive recurrent neural networks* are computationally equivalent to interactive Turing machines with advices [21, 25, 32, 35, 36]. Moreover, the interactive analog and evolving neural networks are universal, in the sense of capturing the computational capabilities of any possible interactive system [32, 36].

Thirdly, the temporal coding approach to neural information processing stripulates that precise spike timing is a significant element in neural coding [20, 165]. Hence, apart from the firing rate of the neural spikes, the *spatiotemporal pattern of discharges* – i.e., ordered and precise interspike interval relationships – are also asumed to be significantly involved in the processing and coding of information in the brain [2, 3, 4, 5, 80, 116, 137, 174, 175, 178, 180]; see also the survey by Villa [176] and the references there. Besides, *attractor dynamics* or *quasi-attractor dynamics* have been associated to perceptions, thoughts and memories, and the *chaotic intinerancy* between those with sequences in thinking, speaking and writing [79, 84, 166, 167, 168]. Specific chaotic attractor dynamics associated to spike series have been ex-

[1] Throughout this manuscript, the concept of "evolvability" is to be understood as a general form of plasticity, i.e., the possibility to change over time.

perimentally observed [39, 40, 177]. Furthermore, the correlation between attractor dynamics and repeating spatiotemporal firing patterns has been observed in simulations of nonlinear dynamical systems [14, 15] as well as in simulations of large scale neuronal networks [77, 78]. Consequently, the spatiotemporal patterns could be the witnesses of an underlying attractor dynamics.

Hence, we initiated the study of the expressive power of recurrent neural networks from the perspective of their attractor dynamics. We considered a neural model where the attractor dynamics of the networks are to the precise phenomena that underly the arising of spatiotemporal patterns of discharges. In this context, we first showed that Boolean recurrent neural networks provided with certain binary type specifications of their attractors are computationally equivalent to Büchi or Muller automata [27, 28, 29]. As a consequence, a novel attractor-based measure of complexity for Boolean neural networks can be deduced. This complexity notably refers to the ability of the networks to perform more or less complicated classification tasks of their input streams via the manifestation of meaningful or spurious attractor dynamics. As an illustration, the attractor-based complexity of a Boolean model of the basal-ganglia thalamocortical network has been computed [29].

As a next step, we generalized the approach to the context of sigmoidal (rather than Boolean) recurrent neural networks [22, 23, 30, 31, 34]. In both deterministic and nondeterministic cases, we proved that the rational-weighted neural networks are equivalent to the Muller Turing machines. The real-weighted and the evolving neural networks are, by contrast, computationally equivalent to each other and strictly more powerful than the Muller Turing machines. For each model, a precise mathematical characterization of its expressive power is provided. In this context also, the complexity of the networks refers to their ability to perform more or less complicated classification tasks of their input streams via the manifestation of meaningful or spurious attractor dynamics.

In the present manuscript, we provide a review of these achievements concerning the computational capabilities of several neural models involved in diverse computational frameworks. Chapters 2 contain the necessary mathematical prerequisites and Chapter 3 recalls the definitions of the classical abstract models of computation. Chapter 4 present the general model of first-order neural networks involved in this work. Chapters 5, 6, and 7 review the main results about the computational capabilities of neural networks involved in classical, interactive, and attractor-based paradigms of computation, respectively. Finally, Chapter 8 furnishes some concluding remarks.

2 PRELIMINARIES

2.1 TOPOLOGY

A *topological space* is a pair $(S, \mathcal{T})$ where S is a set and $\mathcal{T}$ is a collection of subsets of S such that $\emptyset \in \mathcal{T}$, $S \in \mathcal{T}$, and $\mathcal{T}$ is closed under arbitrary unions and finite intersections. The collection $\mathcal{T}$ is called a *topology* on S, and its members are called *open sets*. The *trivial topology* on X is defined by letting only $\emptyset$ and X be open (and hence also closed). By contrast, the *discrete topology* on X is defined by letting every subset of X be open (and hence also closed).

Given some topological space $(S, \mathcal{T})$, the class of *Borel subsets* of S, denoted by $\mathbf{\Delta}^0_1$, is the σ-algebra generated by $\mathcal{T}$, i.e., the smallest collection of subsets of S containing all open sets and closed under countable union and complementation. For every ordinal $\alpha < \omega_1$ (where ω_1 denotes the first uncountable ordinal), one defines the *Borel classes* by transfinite induction:

- $\mathbf{\Sigma}^0_1 = \{X \subseteq S : X \text{ is open}\}$,
- $\mathbf{\Pi}^0_\alpha = \{X \subseteq S : X^\complement \in \mathbf{\Sigma}^0_\alpha\}$,
- $\mathbf{\Sigma}^0_\alpha = \{X \subseteq S : X = \bigcup_{n\geq 0} X_n,\ X_n \in \mathbf{\Pi}^0_{\alpha_n},\ \alpha_n < \alpha,\ n \in \mathbb{N}\}$, for $\alpha > 1$,
- $\mathbf{\Delta}^0_\alpha = \mathbf{\Sigma}^0_\alpha \cap \mathbf{\Pi}^0_\alpha$.

The collection of all classes $\mathbf{\Sigma}^0_\alpha$, $\mathbf{\Pi}^0_\alpha$, and $\mathbf{\Delta}^0_\alpha$ provides a stratification of the whole class of Borel sets known as *the Borel hierarchy*. One has [87, Section 11.B]

$$\mathbf{\Delta}^1_1 = \bigcup_{\alpha<\omega_1} \mathbf{\Sigma}^0_\alpha = \bigcup_{\alpha<\omega_1} \mathbf{\Pi}^0_\alpha.$$

The *rank* of a Borel set $X \subseteq S$ is the smallest ordinal α such that $X \in \mathbf{\Sigma}^0_\alpha \cup \mathbf{\Pi}^0_\alpha \cup \mathbf{\Delta}^0_\alpha$, namely, the minimal number of complementation and countable union operations that are needed to generate X from an initial collection of open sets. It is commonly considered as a relevant measure of the topological complexity of Borel sets.

Given two topological spaces $(S, \mathcal{T})$ and $(S', \mathcal{T}')$, a function $f : S \to S'$ is *continuous* if the preimage by f of any open set (i.e. $\mathbf{\Sigma}^0_1$-set) of S' is an open set (i.e. $\mathbf{\Sigma}^0_1$-set) of S. Consequently, the preimage by f of any $\mathbf{\Sigma}^0_\alpha$-set (resp. $\mathbf{\Pi}^0_\alpha$-set) is a $\mathbf{\Sigma}^0_\alpha$-set (resp. $\mathbf{\Pi}^0_\alpha$-set), for any ordinal $\alpha < \omega_1$. A function $f : S \to S'$ is of *Baire class 1* if the preimage by f of any $\mathbf{\Sigma}^0_1$-set of S' is a $\mathbf{\Sigma}^0_2$-set of S. Hence, the preimage by f of any $\mathbf{\Sigma}^0_\alpha$-set (resp. $\mathbf{\Pi}^0_\alpha$-set) is a $\mathbf{\Sigma}^0_{\alpha+1}$-set (resp. $\mathbf{\Pi}^0_{\alpha+1}$-set), for any ordinal $\alpha < \omega_1$.

2.2 Finite and Infinite Words

Given any set A, we let A^*, A^+, A^n, and A^ω denote respectively the sets of finite sequences, non-empty finite sequences, finite sequences of length n, and infinite sequences of elements of A. We also let $A^{\leq\omega} = A^* \cup A^\omega$ be the set of all possible sequences (finite or infinite) over A. The empty sequence is denoted λ. In this context, the set A is usually called an *alphabet*, and any x of A^* or A^ω is called a finite or infinite *word* over A, respectively, whose every elements are called *letters*. Any set $L \subset A^*$ or $L \subset A^\omega$ is then called a *language* or an ω-language, respectively.

For any word $x \in A^{\leq\omega}$, the *length* of x is denoted by $|x|$, the $(i+1)$-th letter of x will be denoted by $x(i)$ for any $0 \leq i < |x|$, and the subsequence of the n-th first letters of x is denoted by $x[0{:}n]$, with the convention that $x[0{:}0] = \lambda$, the empty word. Hence, any $x \in A^+$ and $y \in A^\omega$ can be written as $x = x(0)x(1)\cdots x(|x|-1)$ and $y = y(0)y(1)y(2)\cdots$, respectively. The fact that x is a *prefix* (resp. *strict prefix*) of y will be denoted by $x \subseteq y$ (resp. $x \subsetneq y$). If $x \subseteq y$, we let $y - x = y(|x|)\cdots y(|y|-1)$ be the *suffix* of y that is not common to x (if $x = y$, then $y - x = \lambda$). The concatenation of x and y is denoted $x \cdot y$ or simply xy, and for any $X \subseteq A^*$ and $Y \subseteq A^* \cup A^\omega$, one sets $X \cdot Y = \{z \in A^* \cup A^\omega : z = x \cdot y \text{ , for some } x \in X \text{ and } y \in Y\}$. A set of the form $\{x\} \cdot A^\omega$ is generally denoted $x \cdot A^\omega$. Sometimes, a concatenated space of the form $A_0 \cdot A_1 \cdot \cdots$ will be naturally identified with the corresponding product space $A_0 \times A_1 \times \cdots$ via the identification $a_0a_1\cdots = (a_0, a_1, \ldots)$.

Given some sequence of finite words $\{x_i : i \in \mathbb{N}\}$ such that $x_i \subseteq x_{i+1}$ for all $i \geq 0$, one defines the *limit* of the x_i's, denoted by $\lim_{i\geq 0} x_i$, as the unique finite or infinite word which is ultimately approached by the sequence of growing prefixes $\{x_i : i \geq 0\}$. Formally, if the sequence $\{x_i : i \in \mathbb{N}\}$ is eventually constant, i.e. there exists an index $i_0 \in \mathbb{N}$ such that $x_j = x_{i_0}$ for all $j \geq i_0$, then $\lim_{i\geq 0} x_i = x_{i_0}$, meaning that $\lim_{i\geq 0} x_i$ corresponds to the smallest finite word containing each word of $\{x_i : i \in \mathbb{N}\}$ as a finite prefix; if the sequence $\{x_i : i \in \mathbb{N}\}$ is not eventually constant, then $\lim_{i\geq 0} x_i$ corresponds to the unique infinite word containing each word of $\{x_i : i \in \mathbb{N}\}$ as a finite prefix.

A function $f : \Sigma^* \to \Sigma^*$ is called *monotone* if the relation $x \subseteq y$ implies $f(x) \subseteq f(y)$, for all $x, y \in \Sigma^*$. It is called *recursive* if it can be computed by some Turing machine. Throughout this paper, any function $\varphi : \Sigma^\omega \to \Sigma^{\leq\omega}$ mapping infinite words to finite or infinite words will be referred to as an *ω-translation*.

The spaces of finite and infinite words of bits $\{0,1\}^*$ and $\{0,1\}^\omega$, respectively, will be of specific importance. The space $\{0,1\}^\omega$ is naturally assumed to be equipped with the product topology of the discrete topology on $\{0,1\}$. Accordingly, the basic open sets are of the form $p \cdot \{0,1\}^\omega$, for some $p \in \{0,1\}^*$. The general open sets are countable unions of basic open sets. This space is Polish (i.e., separable and completely metrizable) [87].

Note that any monotone function $f : \{0,1\}^* \to \{0,1\}^*$ induces "in the limit" an ω-translation $f_\omega : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by

$$f_\omega(x) = \lim_{i\geq 0} f(x[0{:}i])$$

for all $x \in \{0,1\}^\omega$. The monotonicity of f ensures that the value $f_\omega(x)$ is well-defined for all $x \in \{0,1\}^\omega$. In words, the value $f_\omega(x)$ corresponds to the finite or

infinite word that is ultimately approached by the sequence of growing prefixes $\{f(x[0{:}i]) : i \geq 0\}$.

According to these definitions, an ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ will be called *continuous*[1] if there exists a monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$; it will be called *recursive continuous*[2] if there exists a monotone and recursive (i.e. Turing computable) function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$.

Besides, we will also consider the spaces of N-dimensional Boolean, rational and real vectors, denoted by $\mathbb{B}^N$, $\mathbb{Q}^N$ and $\mathbb{R}^N$, respectively. The space $(\mathbb{B}^N)^\omega$ is naturally assumed to be equipped with the product topology of the discrete topology on $\mathbb{B}^N$. Accordingly, the basic open sets are of the form $p \cdot (\mathbb{B}^N)^\omega$, for some $p \in (\mathbb{B}^N)^*$. The general open sets are countable unions of basic open sets. This space is Polish (i.e., separable and completely metrizable) [87]. The spaces $(\mathbb{Q}^N)^\omega$ and $(\mathbb{R}^N)^\omega$ are assumed to be equipped with the product topologies of the usual topologies on $\mathbb{Q}^N$ and $\mathbb{R}^N$, respectively. Accordingly, the basic open sets are of the form $X_0 \cdot \ldots \cdot X_n \cdot (\mathbb{Q}^N)^\omega$ and $X_0 \cdot \ldots \cdot X_n \cdot (\mathbb{R}^N)^\omega$, for some $n \geq 0$, where each X_i is an open set of $\mathbb{Q}^N$ or $\mathbb{R}^N$ for their usual topologies, respectively. Here also, the general open sets are unions of basic open sets. These two spaces are also Polish [87].

In this context, we will use the following two characterizations of *analytic sets* as first projections of either $\mathbf{\Pi^0_2}$-sets or general Borel sets [87]. First, an ω-language $L \subseteq (\mathbb{B}^N)^\omega$ is *analytic* iff there exists some $\mathbf{\Pi^0_2}$-set $X \subseteq (\mathbb{B}^N)^\omega \times \{0,1\}^\omega$ such that $L = \pi_1(X) = \{s \in (\mathbb{B}^N)^\omega : \exists\, e \in \{0,1\}^\omega \text{ s.t. } (s,e) \in X\}$ [87, Exercise 14.3]. Equivalently, $L \subseteq (\mathbb{B}^N)^\omega$ is *analytic* iff there exists some Polish space E and some Borel set $X \subseteq (\mathbb{B}^N)^\omega \times E$ such that $L = \pi_1(X)$ [87, Exercise 14.3]. The class of analytic sets, denoted by $\mathbf{\Sigma^1_1}$, strictly contains that of Borel sets, namely, $\mathbf{\Delta^1_1} \subsetneq \mathbf{\Sigma^1_1}$ [87, Theorem 14.2].

Finally, in the sequel, we will use the word *recursive* to denote a function, language or procedure that is computable by some Turing machine.

[1] The choice of this name comes from the fact that continuous functions over the Cantor space $\mathcal{C} = \{0,1\}^\omega$ can be precisely characterized as limits of monotone functions. We extend this definition in the present broader context of functions from $\{0,1\}^\omega$ to $\{0,1\}^{\leq\omega}$ that can also be expressed as limits of monotone functions.

[2] Our notion of a recursive continuous ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ is a transposition to the present context of the notion of a limit-continuous function $\varphi : \{0,1\}^\omega \to \{0,1\}^\omega$ defined in [98, Definition 12] and [95, Definition 13].

3 AUTOMATA AND TURING MACHINES

In this Chapter, we recall the common abstract models of computation that are finite state automata, stack machines, and Turing machines. We further provide basic facts about Turing machines with advices and oracles.

3.1 AUTOMATA

Definition 1. A *deterministic finite state automaton* is a tuple $\mathcal{A} = (Q, A, i, \delta, \mathcal{F})$ where:

- Q is the finite set of states;
- A is the finite alphabet;
- $i \in Q$ is the initial state;
- $\delta : Q \times A \to Q$ is the partial transition function;
- $\mathcal{F} \subseteq Q$ is the set of final states.

A finite deterministic finite automaton is generally represented as a transition diagram, i.e., a directed labelled graph whose nodes and labelled edges correspond to the states and transitions of the automaton, respectively. Such an automaton is illustrated in Figure 1.

Given some finite state automaton $\mathcal{A} = (Q, A, i, \delta, \mathcal{F})$, every triple (q, a, q') such that $\delta(q, a) = q'$ is called a *transition* of $\mathcal{A}$. Each transition $\delta(q, a) = q'$ signifies that if the automaton is the computational state q and receives input a, then it will move to the next computational state q'. A *path* in $\mathcal{A}$ is a sequence of consecutive transitions

$$\rho = ((q_0, a_1, q_1), (q_1, a_2, q_2), (q_2, a_3, q_3), \ldots)$$

also denoted by

$$\rho : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} q_3 \cdots .$$

The path ρ is said to successively *visit* the states $q_0, q_1, q_2, q_3 \ldots$, and the word $a_1 a_2 a_3 \cdots$ is the *label* of ρ. The state q_0 is called the *origin* of path ρ and ρ is said to be *initial* if its origin is the initial state, namely if $q_0 = i$. If ρ is an infinite path, the set of states visited infinitely many times by ρ is denoted by $\inf(\rho)$.

An automaton $\mathcal{A}$ as defined above is called *deterministic* since it transition function is – by the very definition of being a partial function – deterministic. In other

words, for any state q and input a, there exists at most one state q' such that $\delta(q, a) = q'$. By contrast, an automaton $\mathcal{A}$ is called *nondeterministic* if its fourth component δ consists of a function from $Q \times A$ into $\mathcal{P}(Q)$ (or equivalently, a relation of $Q \times A \times Q$) rather than a partial function from $Q \times A$ into Q. In this case, for any state q and input a, the automaton $\mathcal{A}$ might have several choices for a next computational state to be achieved (corresponding to all the triplets $(q, a, q_1), \ldots, (q, a, q_n) \in \delta$). A deterministic and a nondeterministic automaton are illustrated in Figures 1 and 2, respectively.

Let $\mathcal{A}$ be some deterministic or nondeterministic automaton. A finite initial path ρ of $\mathcal{A}$ is called *successful* if it ends in a final state q, i.e., if the last state visited by ρ belongs to $\mathcal{F}$. A word $u \in A^*$ is said to be *accepted* by $\mathcal{A}$ if it is the label of some successful path. It is *rejected* by $\mathcal{A}$ otherwise. In the deterministic case, this definition means that u is accepted by $\mathcal{A}$ if the unique path induced by u, if it exists, is successful. In the nondeterministic case, u is accepted by $\mathcal{A}$ if there exists some path induced by u which is successful. The set of all words accepted by $\mathcal{A}$ is the *language recognized by* $\mathcal{A}$, denoted by $L(\mathcal{A})$. Deterministic and nondeterministic automata are equivalent in the sense that they recognize the same languages. Examples languages recognized by some deterministic and nondeterministic automaton are provided in Figures 1 and 2, respectively.

The class of languages recognized by finite state automata corresponds precisely to the class of *regular languages*, those that can be expressed by *regular expressions* [71].

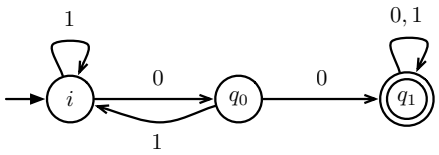


Figure 1 – A deterministic finite state automaton. The nodes correspond to the states of the automaton. The node with an incoming edge is the initial state and the double-circled node is the final state. The labelled edges correspond to the transitions of the automaton: there is an edge from node q to q' labelled by a iff $\delta(q, a) = q'$. This automaton recognizes the set of binary strings which contain 00 as a substring, denoted as A^*00A^* (where $A = \{0, 1\}$).

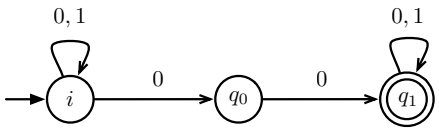


Figure 2 – A nondeterministic finite state automaton which regognizes the same language as that of Figure 1.

3.2 Turing Machines

Definition 2. A *Turing machine* (TM) is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{in}, q_{acc}, q_{rej})$, where

- Q is the finite set of states;
- Σ is the finite input alphabet;
- $\Gamma \supsetneq \Sigma$ is the finite tape alphabet, with $b \in \Gamma \setminus \Sigma$ (b is the blank symbol);
- $\delta : Q \times \Gamma \to Q \times \Sigma \times \{L, R\}$ is the transition function;
- $q_{in} \in Q$ is the initial state;
- $q_{acc}, q_{rej} \in Q$ are the accepting and rejecting states.

A Turing machine is generally represented as in Figure 3. It contains a semi-infinite tape divided into successive cells as well as a read-write head that can read and write symbols from and into these cells and move one cell further to the left or to the right. The behavior of the machine is controlled by its finite programme, i.e., by its transition function.

More precisely, at the beginning of the computation, a finite input $u \in \Sigma^*$ is written on the input tape (each symbol in a cell), the machine is in state q_{in}, and its head is positioned on the first cell of the tape. At each computational step, if the machine is in state $q \in Q$ and its head is currently scanning symbol $a \in \Gamma$, then it will change to state q', replace the symbol a by a', and move its head one cell further in direction d, where q', a', d are given by the relation $\delta(q, a) = (q', a', d)$. A sequence of such consecutive transitions is called a *run* of the machine. A run starting from input u written on the tape is called a *computation* on input u. The computation (or the machine) is said to *halt* if it eventually reaches one of the states q_{acc} or q_{rej}. It is *non-halting* otherwise.

A Turing machine $\mathcal{M}$ as defined above is called *deterministic* since it transition function is deterministic. By contrast, a machine $\mathcal{M}$ is *nondeterministic* if its fourth component δ consists of a function from $Q \times \Gamma$ into $\mathcal{P}(Q \times \Sigma \times \{L, R\})$. In this case, the machine might have several choices for its successive transitions, and in turn, disclose several possible computations for a same input.

In the deterministic case, an input $u \in \Sigma^*$ is *accepted* or *rejected* by a Turing machine $\mathcal{M}$ if the computation of $\mathcal{M}$ on u eventually reaches the state q_{acc} or q_{rej}, respectively; otherwise, the input is neither accepted nor rejected. In the nondeterministic case, the input $u \in \Sigma^*$ is *accepted* by $\mathcal{M}$ if there exists a computation of $\mathcal{M}$ on u which reaches the state q_{acc}; the input is *rejected* by $\mathcal{M}$ if all possible computations of $\mathcal{M}$ on u reach the state q_{rej}; otherwise, the input is neither accepted nor rejected. In Figure 3, the input u written on the tape of the machine $\mathcal{M}$ will be rejected.

The collection of all inputs that are accepted by $\mathcal{M}$ is the *language recognized by* $\mathcal{M}$, denoted $L(\mathcal{M})$. A language L is then called *Turing-recognizable* or *recursively enumerable* if there exists some Turing machine that recognizes it. Furthermore, if the machine $\mathcal{M}$ has the property of halting on every possible inputs – i.e., of being a so-called *decider* – then $L(\mathcal{M})$, the collection of all inputs that are accepted by $\mathcal{M}$, is called the *language decided by* $\mathcal{M}$. A language L is then called *Turing-decidable* or *recursive* if there exists some Turing machine that decides it. Hence, by definition, if L is Turing-decidable, then it is Turing-recognizable. The converse is actually not

true [71]. In Figure 3, Turing machine $\mathcal{M}$ recognizes and decides the language of all binary strings that end with 0.

It can be shown that single-tape Turing machines as defined above are equivalent to multi-tape Turing machines: they recognize and decide the same class of languages. Moreover, the deterministic and nondeterministic Turing machines also recognize and decide the same class of languages [71].

If $\mathcal{M}$ is a deterministic Turing machine that halts on all inputs (a decider), the *time complexity* of $\mathcal{M}$ is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of steps that $\mathcal{M}$ uses on any input of length n. If $\mathcal{M}$ is a nondeterministic, its *time complexity* is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of steps that $\mathcal{M}$ uses on any possible computation of every input of length n. If $f(n) = \mathcal{O}(p(n))$ for some polynomial p, we say that the time complexity is polynomial. The classes of languages decidable in polynomial time by deterministic and nondeterministic Turing machines are denoted by **P** and **NP**, respectively. The inclusion **P** $\subseteq$ **NP** holds by definition; the issue of knowing whether this inclusion is strict or not remains an open problem, a most important one in theoretical computer science.

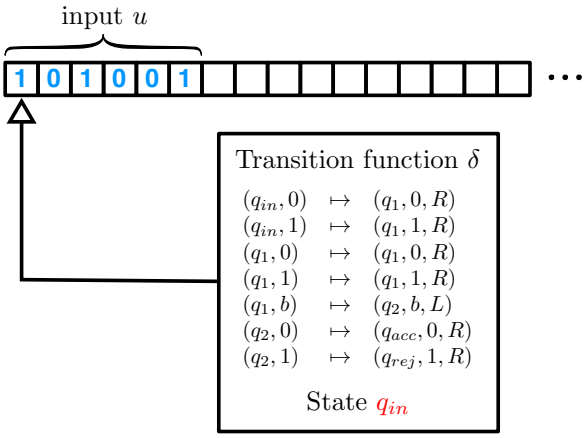


Figure 3 – A deterministic Turing machine $\mathcal{M}$ which decides the language of all binary strings that end with 0, i.e., $L(\mathcal{M}) = A^*0$. At the beginning of the computation, an input u is written on the tape. Then, the machine follows the transitions given by δ and eventually halts in either the accepting state q_{acc} or the rejecting state q_{rej}.

Finally, note that the Turing machine paradigm of computation can naturally be transposed from the context of "languages" to that of "functions". In this case, we suppose that Turing machines are equipped with a single halting state q_{halt} instead of the two q_{acc} and q_{rej}. Accordingly, a partial function $f : \Sigma^* \to \Sigma^*$ is said to be *Turing-computable* or *partial-recursive* if there exists some Turing machine $\mathcal{M}$ such that, on any input $u \in Dom(f)$, $\mathcal{M}$ halts with $f(u)$ written on its tape, and on any input $u \notin Dom(f)$, $\mathcal{M}$ doesn't halt. A total function $f : \Sigma^* \to \Sigma^*$ is called *recursive* if there exists some Turing machine $\mathcal{M}$ such that, on any input u, $\mathcal{M}$ halts with $f(u)$ written on its tape.

3.3 Stack Machines

A *p-stack machine* $\mathcal{M}$ is an automaton provided with p binary stacks. The automaton has the ability either to push 0, push 1, or pop the top element of each stack. At each computational step, if the machine is in state $q \in Q$, receives input bit $x \in \{0,1\}$, and has as top elements ifs stacks $t_0, \ldots, t_{p-1} \in \{0,1,b\}$ (where b is an blank symbol meaning that the stack is empty), then it will move to state $q' \in Q$ and either push 0, or push 1, or pop the top element of each stack. Hence, the transition function of $\mathcal{M}$ is of the form

$$\delta : Q \times \{0,1\} \times (\{0,1\}^*)^p \to Q \times (\{0,1\}^*)^p$$

where the transformation of each content's stack is the result of a "push 0", "push 1", or "pop" operation. At the beginning of the computation, all stacks are empty. As usual, an input u is accepted or rejected by $\mathcal{M}$ if the computation of $\mathcal{M}$ on u end in an accepting or rejecting state, respectively.

In the sequel, we will use the fact that p-stack machines with $p \geq 2$ are computationally equivalent to Turing machines [71]. Indeed, on the one hand, every p-stack machine $\mathcal{S}$ can obviously be simulated by some p-tape Turing machine $\mathcal{M}$: each stack of $\mathcal{S}$ is represented by a corresponding tape of $\mathcal{M}$. Conversely, every single tape Turing machine $\mathcal{M}$ can be simulated by some 2-stack machine $\mathcal{S}$: the idea of the simulation is to cut the tape of $\mathcal{M}$ in half at the location of its read-write head, and then, register the content of each half of the tape into some corresponding stack of $\mathcal{S}$. In other words, $\mathcal{S}$ is designed such that, at each computational step, the contents of its two stacks corresponds precisely to the "left part" and "right part" of the tape of $\mathcal{M}$.

3.4 Turing Machines with Oracles and Advices

In complexity theory, oracle Turing machines are used to characterize classes of complexity that are beyond the scope of the classical Turing machine model [127].

An *oracle Turing machine* (TM/O) consists of a 2-tape Turing machine $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{in}, q_{acc}, q_{rej})$ together with an oracle function $\alpha : \Sigma^* \to \Sigma^*$ and a designated oracle state $q_{oracle} \in Q$. The first and second tapes are the work and the oracle tape, respectively. At the beginning of the computation, a finite input $u \in \Sigma^*$ is written on the work tape, the oracle tape is empty, the machine is in state q_{in}, and its two heads are positioned on the first cells of each tape. Along the computation, every time the machine reaches the oracle state q_{oracle}, it makes a so-called extra-recursive call to its oracle and has the word $\alpha(w)$ being written instantaneously on its oracle tape, where w is the current content of the work tape. With the use of the oracle functions that are non-recursive, oracle Turing machines are obviously strictly more powerful than classical Turing machines. An oracle Turing machine is illustrated in Figure 4.

In this work, we will focus on specific oracle Turing machines whose oracles are uniform for all inputs of the same length. These are the Turing machines with advice. Formally, a *Turing machine with advice* (TM/A) consists of a 2-tape Turing machine $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{in}, q_{acc}, q_{rej})$ together with a function $\alpha : \mathbb{N} \to \Sigma^*$. Note

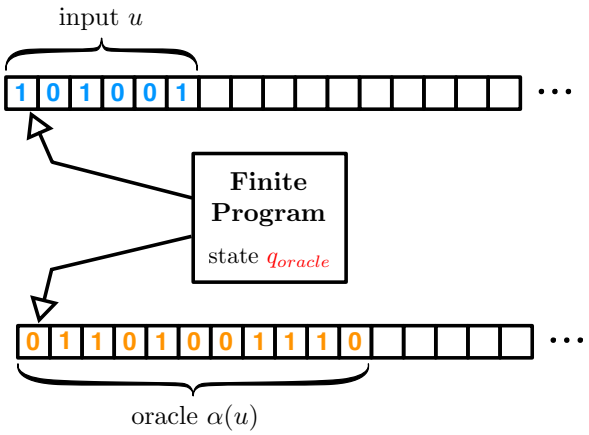


Figure 4 – An oracle Turing machine.

that the domain of the advice function is now restricted to $\mathbb{N}$. The first and second tapes are the work and the advice tape, respectively. At the beginning of the computation, a finite input $u \in \Sigma^*$ of length n is written on the work tape, the advice tape is empty, the machine is in state q_{in}, and its two heads are positioned on the first cells of each tape. Then, at the first computational step (and only at this time), by means of a so-called extra-recursive call, the machine has the advice word $\alpha(n)$ being written on its advice tape. Consequently, the same word $\alpha(n)$ is queried for all inputs of length n. Afterwards, the machine continues its computation in the classical way, with the two input and advice words written on its two tapes. A Turing machine with advice is illustrated in Figure 4.

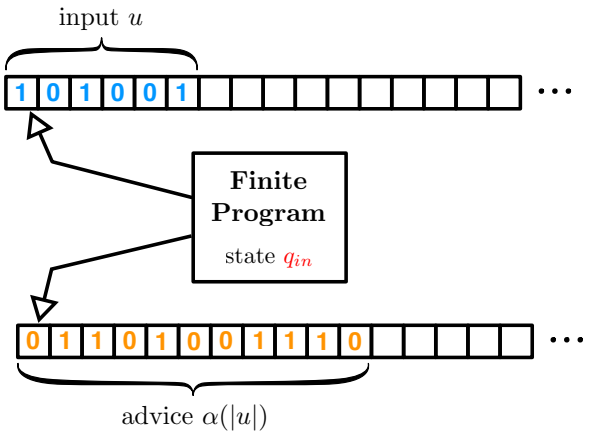


Figure 5 – A Turing machine with advice.

It can easily be show that Turing machines with advice of exponential length are capable of deciding all possible languages. Hence, of specific interest are the Turing machines with advice whose advices are not large, i.e., polynomially bounded. A *Turing machine with polynomially bounded advice* (TM/poly(A)) is a Turing machine

with advice $\mathcal{M}$ whose advice function α satisfies $n \mapsto |\alpha(n)| \in \mathcal{O}(p(n))$, for some polynomial p. The class of languages decidable in polynomial time by Turing machines with polynomially bounded advice is denoted by **P/poly** (the "**P**" refers the the polynomial time of computation and the "**poly**" to the polynomially-bounded length of the advice). It is known that Turing machines with polynomially bounded advice are strictly more powerful that classical Turing machine. In fact, $\mathbf{P} \subsetneq \mathbf{P/poly}$ and **P/poly** even contains non-recursive languages [16].

4 Recurrent Neural Networks

4.1 Generalities

In this work, a *neural network* consists of a finite set of entities – the *neurons* or *cells* – interconnected together in a directed way by means of *synaptic connections*. It can be depicted as a graph whose nodes and directed labelled edges represent the neurons and weighted synaptic connections between those, respectively. The *processing of information* consists of a flow of values that propagates throughout the neurons via their synaptic connections. This issue will be more precise in the sequel. A neural network is illustrated in Figure 6.

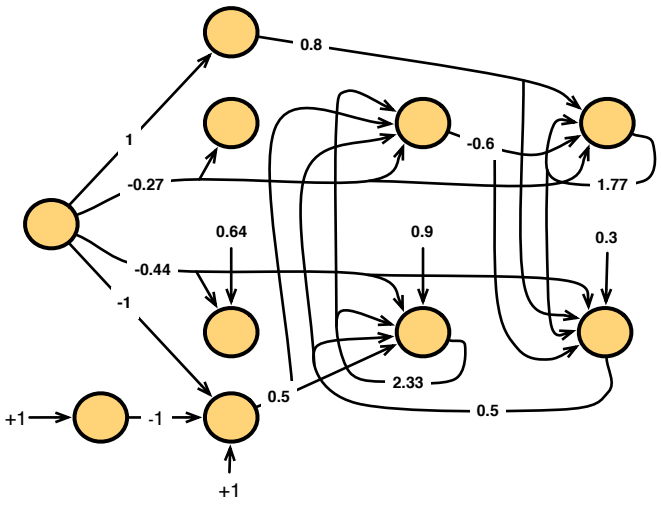


Figure 6 – A neural network. The nodes and labelled directed edges represent the neurons and weighted synaptic connections, respectively.

Artificial neural networks are particularly appropriate for the implementation of *learning algorithms*, i.e., algorithms that infer some input-output correlation from existing data, rather than having it explicitly implemented by static program instructions. In artificial neural networks, learning is achieved via successive modifications of the neural architecture, usually the synaptic weights, with the purpose of building a suitable model for the task to be performed. Nowadays, deep learning methods for neural networks represents a highly active field of research. For a detailed overview, see [143] and the references therein.

A neural network is called *feedforward* if its architecture doesn't contain any loop, i.e., if its corresponding directed graph is acyclic. Feedforward neural networks were the first and simplest type of artificial neural networks considered. In these latter, the information is processed in only one direction: forward. Such neural networks have an extensive range of applications. Their architecture turns out to be particularly suitable for the implementation of learning methods, and in particular, for the famous class of backpropagation algorithms [139, 142, 143]. In this case, the synaptic weights of the network are updated by backward induction according to some gradient descent process. From a theoretical perspective, feedforward neural networks turn out to have universal approximation capabilities: they can approximate any well-behaved continuous or Borel real function to any degree of accuracy [74, 75].

A neural network is called *recurrent* if its architecture contains some loop, i.e., if its corresponding directed graph is cyclic. The dynamics of these networks is intrinsically more complex than that of feedforward nets. In fact, the information might be recurrently processed forever – by entering into some cycle of the graph – thus creating some internal memory of the network. In their seminal theoretical study concerning the capabilities of neural networks, McCulloch and Pitts already distinguished the cases of feedforward and recurrent neural networks [118]. They write: "The treatment of nets which do not satisfy our previous assumption of freedom of circle is very much more difficult." Nowadays, recurrent neural networks can be applied to tasks such as unsegmented connected handwriting recognition or speech recognition. From a theoretical perspective, their computational power is drastically more important than that of feedforward nets, for they can implement some notion of memory that is impossible to be achieved in feedforward nets.

4.2 First-Order Recurrent Neural Networks

In general, the *dynamics* of a neural network refers to the successive values that the neurons of the network will take – their *activation values* – as time elapses. These activation values are computed by means of an *activation function* associated to each neuron. Throughout this work, time is assumed to be discretized and to elapse by steps of 1. Moreover, the neurons are assumed to update their activation values in a *synchronous* way – i.e., all cells are updated at the same time – at each discrete time step. These updating procedures can be of various kinds. Here, we will exclusively focus on *first-order* recurrent neural networks. In this case, the activation values of every neuron are updated by means of weighted sums of other cells' activation values and inputs. By contrast, in higher-order recurrent neural networks, the activation values of every cell are computed by means of polynomials of other cells' activation values and inputs. The degree of the polynomial represents the order of the network.

Formally, a *first-order recurrent neural network* (RNN) consists of a synchronous network of neurons related together in a general architecture. The network contains N internal neurons $(x_i)_{i=1}^N$, M parallel input neurons $(u_i)_{i=1}^M$, and P designated output neurons $(x_{i_j})_{j=1}^P$ chosen among the N. At each time step, the activation value of every neuron is updated by applying a function to some weighted affine combi-

nation of the activation values of other internal and input neurons at the previous time step. Besides, in our work, the synaptic weights will often be supposed to be time dependent. Hence, given the activation values of the internal and input cells $(x_j)_{j=1}^N$ and $(u_j)_{j=1}^M$ at time t, the activation value of each cell x_i at time $t+1$ is updated by the following equation:

$$x_i(t+1) = f\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t)\cdot u_j(t) + c_i(t)\right) \text{ for } i = 1,\ldots,N \quad (4.1)$$

where the $a_{ij}(t)$ and $b_{ij}(t)$ are bounded and time dependent synaptic weights, the $c_i(t)$ are bounded and time dependent biases for the cells x_i, and f is the activation function of the neurons, which can be of two following kinds:

- the *Heaviside* step function, usually denoted by θ, and defined by

$$\theta(x) = \begin{cases} 0 & \text{if } x < 1, \\ 1 & \text{if } x \geq 1; \end{cases}$$

- the *linear-sigmoid* function, usually denoted by σ, and defined by

$$\sigma(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } 0 \leq x \leq 1, \\ 1 & \text{if } x > 1. \end{cases}$$

The dynamics of a neuron given by Equation (4.1) and provided with a Heaviside or a linear-sigmoid activation function are illustrated in Figure 7.

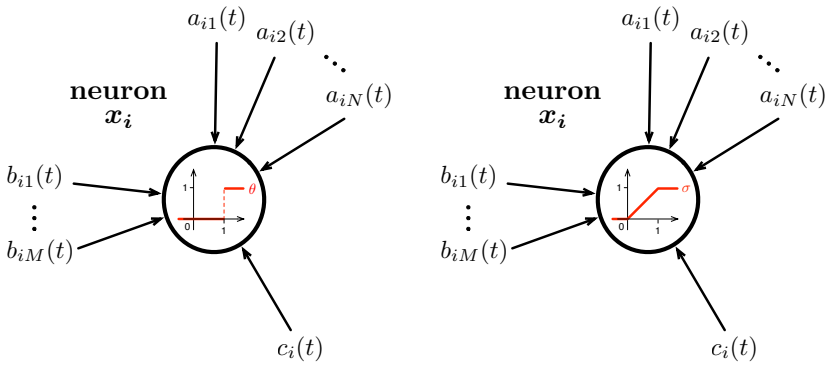


Figure 7 – Illustration of the dynamics of a neuron given by Equation (4.1) in the case of the activation function being given by the Heaviside function θ or the linear-sigmoid function σ, respectively.

The time dependence of the synaptic weights captures the evolving capabilities of the network. The boundedness condition expresses the fact that the synaptic strengths are confined into a certain range of values imposed by the biological constitution of the neurons. It formally states that there exist a lower and an upper bound S and S' such that $a_{ij}(t), b_{ij}(t), c_i(t) \in [S, S']$ for every $t \geq 0$.

In the sequel, a neuron will be called *Boolean* if its activation values are always 0 or 1. A network whose all neurons are Boolean is also called *Boolean*. In particular, a network whose activation functions of every neurons are Heaviside is necessarily Boolean. A neuron will be called *sigmoidal* if its activation function is the linear-sigmoid one. A network which contains at least one sigmoidal cell is called *sigmoidal*.

Moreover, a synaptic weight w will be called *static* if $w(t) = C$, for all $t \geq 0$. It is *bi-valued evolving* if $w(t) \in \{0, 1\}$, for all $t \geq 0$, and it is *(general) evolving* if $w(t) \in [S, S']$, for all $t \geq 0$. Note that static and bi-valued evolving weights are particular cases of general evolving weights.

Consider some RNN $\mathcal{N}$ provided with M Boolean input cells and N internal cells. For each time step $t \geq 0$, the Boolean vectors

$$\boldsymbol{u}(t) = (u_1(t), \ldots, u_M(t)) \text{ and } \boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))$$

describing the activation values of the M input and N internal units of $\mathcal{N}$ at time t are the *input* and *state* of $\mathcal{N}$ at time t. Assuming the initial state of the network to be $\boldsymbol{x}(0) = \mathbf{0}$, any *input stream*

$$s = \boldsymbol{u}(0)\boldsymbol{u}(1)\cdots\boldsymbol{u}(n-1)$$

induces via Equation (4.1) a sequence of consecutive states

$$c_s = \boldsymbol{x}(0)\boldsymbol{x}(1)\cdots\boldsymbol{x}(n).$$

called the *computation* induced by s.

The dynamics of a simple recurrent neural network is illustrated below.

Example 3. Consider the Boolean recurrent neural network $\mathcal{N}$ whose dynamics is governed by the following system of equations:

$$\begin{pmatrix} x_1(t+1) \\ x_2(t+1) \\ x_3(t+1) \end{pmatrix} = f\left[\begin{pmatrix} 0 & -\frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}\right] \tag{4.2}$$

The network $\mathcal{N}$ is depicted in Figure 8.

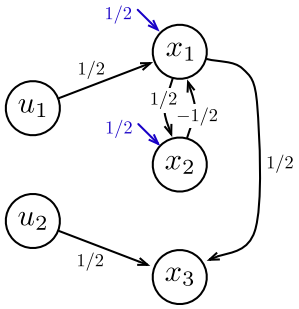


Figure 8 – A simple Boolean neural network with two input units (u_1, u_2) and three internal cells (x_1, x_2, x_3).

If the activation function f of Equation (4.2) is the Heaviside function θ, the network is Boolean. In this case, if we assume that the initial state of the network is $\boldsymbol{x}(0) = \mathbf{0}$, then the input stream

$$s = \binom{0}{0}\binom{1}{0}\binom{0}{1}\binom{0}{0}$$

induces the corresponding sequences of states

$$c_s = \binom{0}{0}{0}\binom{0}{0}{0}\binom{1}{0}{0}\binom{0}{1}{1}\binom{0}{1}{1}.$$

If the activation function f of Equation (4.2) is the linear-sigmoid function σ, the network is not anymore Boolean but sigmoidal. In this case, if we assume that the initial state of the network is $\boldsymbol{x}(0) = \mathbf{0}$, then the input stream s induces the sequence of states

$$c_s = \binom{0}{0}{0}\binom{0}{0}{0}\binom{1}{1/2}{0}\binom{1/4}{1}{1}\binom{0}{5/8}{1/8}.$$

5 CLASSICAL COMPUTATION

5.1 INTRODUCTION

We assume that at least some aspects of the brain processes are of a classical computational nature, along the lines defined by Turing [171], namely, as black box function-based transformation of a given input into some corresponding output.[1] But if the brain computes, it certainly does it differently than today's computers. In particular, the brain admits a highly flexible neural architecture. Biological mechanisms like synaptic plasticity, cell birth and death, and changes in connectivity are intimately related to the storage and encoding of memory traces in the central nervous system. They provide the basis for most models of learning and memory in neural networks [1, 38, 42, 117, 138].

More precisely, the embryonic nervous system is initially driven by genetic programs that control neural stem cell proliferation, differentiation, and migration, via the actions of a limited set of trophic factors and guidance cues. After a relatively short period of stable synaptic density, a pruning process begins: synapses are constantly removed, yielding a marked decrease in synaptic density due to apoptosis – genetically programmed cell death – and selective axon pruning [81]. Overproduction of a critical mass of synapses in each cortical area may be essential for their parallel emergence through competitive interactions between extrinsic afferent projections [41]. Background activity and selected patterns of afferent activity are likely to shape deeply the emergent circuit wiring [147]. Synapses can change their strength in response to the activity of both pre- and post-synaptic cells following spike timing dependent plasticity (STDP) rules [138]. Developmental and/or learning processes are likely to potentiate or weaken certain pathways through the network by affecting the number or efficacy of synaptic interactions between the neurons.

In the context of AI, the consideration of evolving neural architectures in so-called Evolving Connectionist Systems (ECoS) has proven to be fruitful and significantly increased in applications [85, 184].

From a theoretical perspective, the general issue of evolvability and plasticity is difficult to handle in brain's modelling, and has generally been neglected in the classical literature concerning the computational capabilities of brain-like models (see e.g. [89, 118, 119, 120, 123, 151, 153, 154, 155, 156]). Hence, the following questions naturally arise: Can we approach the issue of the brain's capabilities

[1] We do not enter into the philosophical considerations about the meaning of "computing" and the justification that the brain "computes" in a relevant sense.

from a non-static perspective? Can we understand and characterize the computational capabilities of a neural model incorporating the crucial feature of plasticity or evolvability?

We answered positively by providing a detailed study of the computational capabilities of an evolving neural model, where the networks can update their architectures at each discrete time step [24, 26]. Our study focuses on the mere concept of evolvability of the model, with no assumption on the particular environment, so that the nature of the updates is assumed to be not constrained. In this general context, we show that the *evolving recurrent neural networks* are computationally equivalent to Turing machines with polynomially bounded advices, and accordingly, achieve a *super-Turing* computational power, equivalent to that of analog recurrent neural networks [149, 154]. These capabilities are attained irrespective of whether the synaptic weights of the networks are modelled by rational or real numbers, and moreover, irrespective of whether the patterns of evolvability are restricted to bi-valued updates (namely changes between two possible distinct synaptic weights, like 0 and 1) or expressed by any other more general form of updating.

Besides, a natural question to be addressed concerns the robustness of the super-Turing computational power of these evolving networks when subjected to various kinds of noise. It has been shown that the presence of analog noise would generally strongly decrease the computational power of the underlying systems [18, 111, 115], whereas the consideration of some discrete source of stochasticity would rather tend to increase or maintain the capabilities of the systems [153]. We showed that the evolving neural network models falls under the scope of these results [26]. More precisely, on the one hand, both evolving rational and evolving real recurrent neural networks have their computational power decreased to regular or definite languages in the presence of analog noise, as described in [111, 115], respectively. On the other hand, the evolving networks have their capabilities maintained to the super-Turing level in the presence of some discrete source of stochasticity, as presented in [153].

Overall, these results allow to drop any kind of analog assumption and replace it by the more biological concept of evolvability in neural network models. They support the claim that the general mechanism of evolvability is crucially involved in the computational and dynamical capabilities of biological neural networks. In this sense, they provide a theoretical complement to the numerous experimental studies about the importance of the general mechanism of evolvability in the brain's information processing [1, 48, 69].

In this Chapter, we review the main results concerning the computational power of recurrent neural networks. Section 5.2 concerns the correspondence between Boolean recurrent neural networks and finite state automata [89, 119]. Sections 5.3 and 5.4 concern the computational equivalence between rational-weighted or real-weighted recurrent neural networks and Turing machines or Turing machines with advices, respectively [154, 155]. Section 5.5 presents in details the computational capabilities of evolving recurrent neural networks [24, 26]. Section 5.6 provides some concluding remarks.

5.2 Boolean Neural Networks

It has early been observed that Boolean recurrent neural networks are computationally equivalent to finite state automata [89, 119]. More precisely, recurrent neural networks composed of McCulloch and Piits's cells, the Boolean neurons described in Chapter 4, can simulate and be simulated by finite state automata. In Minsky's own words:

> It is evident that each neural network of the kind we have been considering is a finite-state machine. [...] It is interesting and even surprising that there is a converse to this. Every finite-state machine is equivalent to, and can be "simulated" by, some neural net [119].

Here, a Boolean recurrent neural network (B-RNN) is a neural network as defined in Chapter 4, i.e., with M Boolean input cells, N internal cells, and a dynamics governed by the following equation:

$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t) \cdot u_j(t) + c_i(t)\right) \text{ for } i = 1, \ldots, N \quad (5.1)$$

where θ is the Heaviside step activation function. We suppose that the weights and bias can be indifferently modelled by rational or real numbers. The following results are not affected by this condition.

The concept of language recognition can be straightforwardly adapted to the context of Boolean neural networks. The idea is to consider that certain state(s) of the network are final, and to define that some input stream is *accepted* or *rejected* by the network if this latter ends in a final or a non-final state after having processed input u, respectively. The language *decided* by the network consists of the set of input streams that are accepted by it. A language is *decidable* by some Boolean neural net if it can be decided by such a network. According to these considerations, the computational equivalence between Boolean recurrent neural networks and finite state automata [119] can be stated as follows.

Theorem 4. *Let $L \subseteq (\mathbb{B}^M)^*$ be some language. The following conditions are equivalent:*

(a) L is decidable by some Boolean recurrent neural network;

(b) L is decidable by some finite state automaton;

(c) L is regular.

We provide a sketch of the proof of this result. First, note that the equivalence between conditions (b) and (c) is well-known in automata theory [71].

For the implication $(a) \to (b)$, we show that any Boolean recurrent neural network can be simulated by some finite state automaton. Given some network $\mathcal{N}$, one constructs a corresponding automaton $\mathcal{A}$ whose nodes correspond to the states of $\mathcal{N}$, and such that there is an edge from node $\boldsymbol{s}$ to $\boldsymbol{s}'$ labelled by $\boldsymbol{x}$ in $\mathcal{A}$ if and only if the network $\mathcal{N}$ moves from state $\boldsymbol{s}$ to $\boldsymbol{s}'$ when it receives the Boolean input vector $\boldsymbol{x}$. Since $\mathcal{N}$ contains finitely many cells, say N, then it contains also finitely many

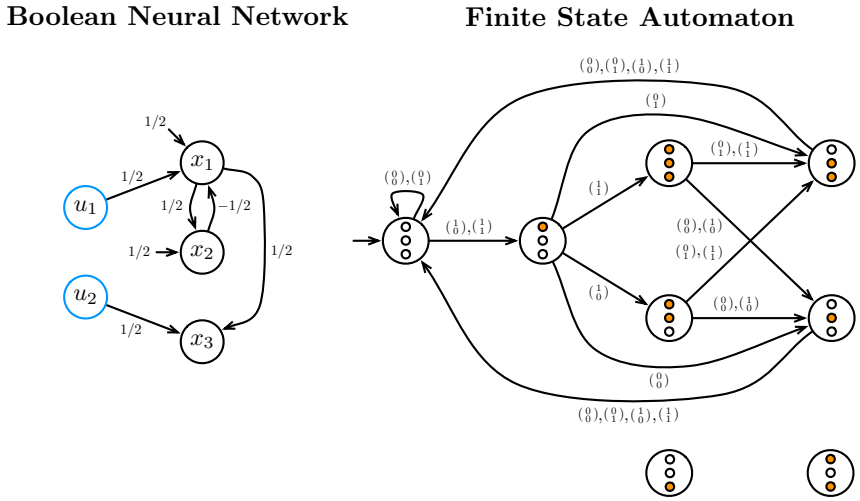


Figure 9 – Translation from a Boolean recurrent neural network $\mathcal{N}$ to an equivalent finite state automaton $\mathcal{A}$. The nodes of $\mathcal{A}$ are the different states of $\mathcal{N}$. They are represented as colored triple dots that depict the three internal cells of $\mathcal{N}$: a white or a colored dot depicts a quiet or a spiking cell. Moreover, there is an edge from node s to node s' labelled by x in $\mathcal{A}$ if and only if the network $\mathcal{N}$ moves from state s to s' when it receives input x.

states, namely 2^N, and therefore, $\mathcal{A}$ contains finitely many nodes, i.e., it is a finite state automaton. This construction is illustrated in Figure 9.

Concerning the converse implication $(b) \to (a)$, we show that any finite state automaton can be simulated by some Boolean recurrent neural network. Given some automaton $\mathcal{A}$ working on alphabet A, one constructs a network $\mathcal{N}$ which contains sufficiently many input cells to encode all letters of A, and whose internal cells are oganized in a grid manner: each row of cells correspond to some specific letter of A and each column of cells correspond to some state of $\mathcal{A}$. Then, it is possible to design the weighted synaptic connections between the cells such that the cell of location (i, j) in the grid will spike if and only if the automaton $\mathcal{A}$ had previously received the i-th letter of alphabet A and is currently in the j-th computational state. In this way, for any input u, one has a biunivocal correspondence between the successive states visited by $\mathcal{A}$ and the successive spiking configurations of $\mathcal{N}$. The automaton $\mathcal{A}$ is therefore perfectly simulated by the network $\mathcal{N}$. This construction is illustrated in Figure 10.

5.3 Rational-Weighted Neural Networks

The study of the computational power of recurrent neural networks requires the consideration of a specific model of RNNs capable to perform computation and decision of formal languages. In his way, a mathematical comparison with the languages computed by classical abstract models of computation, like Turing machines and Turing machines with advice in our case, becomes possible.

For this purpose, we consider a notion of formal recurrent neural networks

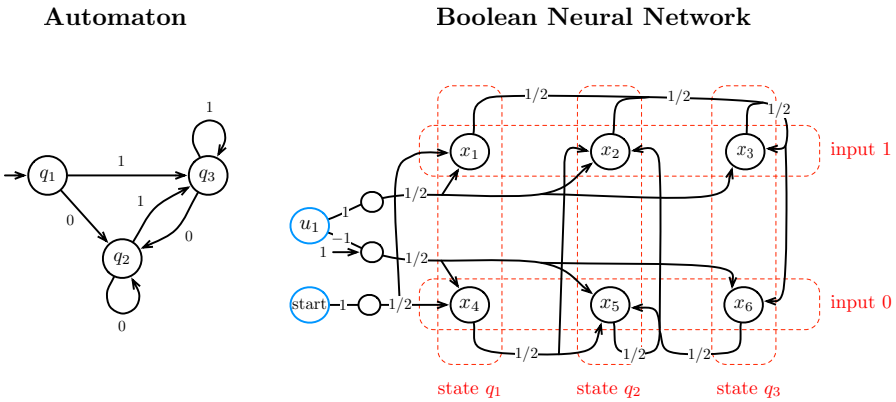


Figure 10 – Translation from a finite state automaton $\mathcal{A}$ to an equivalent Boolean recurrent neural network $\mathcal{N}$. The network $\mathcal{N}$ contains sufficiently many input cells to encode all letters of A: in this case, one Boolean input cell is sufficient to encode the two possible letters 0 and 1. The internal cells of $\mathcal{N}$ are oganized in a grid manner: each row of cells correspond to some specific letter of alphabet A (two rows here, for the letters 0 and 1) and each column of cells correspond to some state of $\mathcal{A}$ (three columns here, for the states q_1, q_2, and q_3). The weighted synaptic connections are designed such that the cell of location (i, j) in the grid will spike if and only if the automaton $\mathcal{A}$ had previously received the i-th letter of alphabet A and is currently in the j-th computational state. This pattern is realized with a time delay of 2. The start cell spikes only at time $t = 0$.

which adheres to a rigid encoding of the way binary strings are processed as input and output between the network and the environment, as presented in [154, 155]. A *formal recurrent neural network* is a recurrent neural network, as described in Chapter 4, provided with two Boolean input cells u_d and u_v, N internal sigmoidal cells $(x_i)_{i=1}^N$, and two Boolean output cells y_d and y_v. The so-called data lines u_d and y_d carry some uninterrupted incoming and outgoing binary data, respectively; the so-called validation lines u_v and y_v take value 1 to indicate when their corresponding data line is active and take value 0 otherwise. Moreover, the networks are assumed to be designed in such a way that the two output cells y_d and y_v are not fed into any other cell.

The dynamics of the network is computed in the usual way: given the activation values of the inputs u_d and u_v and the internal neurons $(x_j)_{j=1}^N$ at time t, the activation values of each internal and output neuron x_i, y_d and y_v at time $t+1$ are updated by the following equations, respectively:

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j\in\{d,v\}} b_{ij}(t)\cdot u_j(t) + c_i(t)\right) \quad \text{for } i = 1,\ldots,N \tag{5.2}$$

$$y_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j\in\{d,v\}} b_{ij}(t)\cdot u(t) + c_i(t)\right) \quad \text{for } i = d, v \tag{5.3}$$

where $a_{ij}(t)$, $b_i(t)$, and $c_i(t)$ are the weights of the synaptic connections and the bias of the network at time t, and σ and θ are the linear-sigmoid and Heaviside step

activation functions, respectively. From this point onwards, all considered neural networks will be assumed to be of that form. A formal recurrent neural network is illustrated in Figure 11.

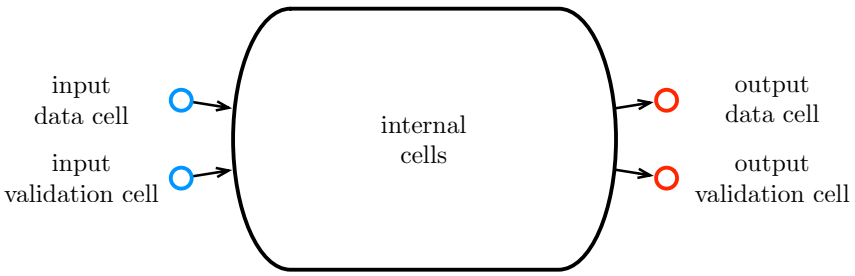


Figure 11 – Schematic representation of a formal recurrent neural network.

The formal RNNs perform computation over finite input strings of bits as follows: given some formal RNN $\mathcal{N}$ and some input string $u = u_0 \cdots u_k \in \{0,1\}^+$, we say that u is *classified in time* τ by $\mathcal{N}$ if given the input streams

$$\begin{aligned} u_d(0)u_d(1)u_d(2)\cdots &= u_0 \cdots u_k 000 \cdots \\ u_v(0)u_v(1)u_v(2)\cdots &= \underbrace{1\cdots 1}_{k+1}000\cdots \end{aligned}$$

the network $\mathcal{N}$ produces the corresponding output streams

$$\begin{aligned} y_d(0)y_d(1)y_d(2)\cdots &= \underbrace{0\cdots 0}_{\tau-1}\eta_u 000\cdots \\ y_v(0)y_v(1)y_v(2)\cdots &= \underbrace{0\cdots 0}_{\tau-1}1000\cdots \end{aligned}$$

where $\eta_u \in \{0,1\}$. The input string u is said to be *accepted* or *rejected* by $\mathcal{N}$ if $\eta_u = 1$ or $\eta_u = 0$, respectively. Moreover, for any non-decreasing function $f : \mathbb{N}^* \to \mathbb{N}^*$ and any language $L \subseteq \{0,1\}^+$, we say that L is *decided* by $\mathcal{N}$ in time f if and only if every string $u \in \{0,1\}^+$ is classified by $\mathcal{N}$ in time $\tau \leq f(|u|)$, and $u \in L \Leftrightarrow \eta_u = 1$. Finally, a given language L is then said to be *decidable in time* f by some network if and only if there exists a RNN $\mathcal{N}$ that decides L in time f. A language L is simply said to be *decidable* by some network if and only if there exist a RNN $\mathcal{N}$ and a non-decreasing function $f : \mathbb{N}^* \to \mathbb{N}^*$ such that $\mathcal{N}$ decides L in time f.

In this section, we consider that the synaptic weights and bias of the network are static and modelled by rational numbers.[2] The corresponding networks are called *static rational recurrent neural networks*, and denoted by St-RNN[$\mathbb{Q}$]s.

In this context, Siegelmann and Sontag proved that first-order rational recurrent neural networks involving only finitely many cells and simple short rational weights are Turing equivalent [151, 155]. In fact, on the one hand, any function determined by Equations (5.2) and (5.3) and involving static rational weights is necessarily recursive, i.e., computable by some Turing machine. On the other hand,

[2] We recall that a weight $w(t)$ is static and rational if $w(t) = C \in \mathbb{Q}$, for all $t \geq 0$.

any Turing machine can be simulated in real time by some static rational recurrent neural network. The result can be expressed as follows.

Theorem 5. *St-RNN[$\mathbb{Q}$]s are Turing equivalent. More precisely, a language L is decidable by some St-RNN[$\mathbb{Q}$] if and only if L is decidable by some TM, i.e., iff L is recursive.*

For sake of completeness, we provide a sketch of the proof of this result. First, as already mentioned, any St-RNN[Q] $\mathcal{N}$ has its dynamics governed by Equations (5.2) and (5.3), which in the case of rational weights, is obviously recursive. Consequently, $\mathcal{N}$ can be simulated by some Turing machine $\mathcal{M}$.

Conversely, consider some Turing machine $\mathcal{M}$. Then $\mathcal{M}$ is computationally equivalently to some p-stack machine $\mathcal{S}$ with $p \geq 2$ (See Section 3.3). We show that $\mathcal{S}$ can be simulated by some St-RNN[$\mathbb{Q}$] $\mathcal{N}$.

Towards this purpose, we encode every stack content $w = w_0 \cdots w_{n-1} \in \{0,1\}^*$ as the rational number $q_w = \sum_{i=0}^{n-1} \frac{2 \cdot w(i)+1}{4^i} \in [0,1]$. For instance, $w = 1110$ is encoded into $q_w = \frac{3}{4} + \frac{3}{16} + \frac{3}{64} + \frac{1}{256}$. According to this encoding, the required stack operations can be performed by simple functions involving the sigmoid-linear function σ, as described above:

- Reading the top of the stack: $top(q_w) = \sigma(4q_w - 2)$
- Pushing 0 into the stack: $push_0(q_w) = \sigma(\frac{1}{4}q_w + \frac{1}{4})$
- Pushing 1 into the stack: $push_1(q_w) = \sigma(\frac{1}{4}q_w + \frac{3}{4})$
- Popping the stack: $pop(q_w) = \sigma(4q_w - (2top(q_w) + 1))$
- Emptiness of the stack: $empty(q_w) = \sigma(4q_w)$

For instance, if $w = 1110$, then $q_w = \frac{3}{4} + \frac{3}{16} + \frac{3}{64} + \frac{1}{256}$ and one has:

- $top(q_w) = 1$
- $push_0(q_w) = \frac{1}{4} + \frac{3}{16} + \frac{3}{64} + \frac{3}{256} + \frac{1}{1024}$
- $push_1(q_w) = \frac{3}{4} + \frac{3}{16} + \frac{3}{64} + \frac{3}{256} + \frac{1}{1024}$
- $pop(q_w) = \frac{3}{4} + \frac{3}{16} + \frac{1}{64}$
- $empty(q_w) = 1$ (meaning that w is not empty).

Consequently, one can store the encoding of each stack q_w as the rational activation value of a neuron, and every stack operation on q_w can be performed by some simple neural circuit implementing the corresponding function.

Based on these considerations, one can design a St-RNN[Q] $\mathcal{N}$ which correctly simulates the p-stack machine $\mathcal{S}$. The network $\mathcal{N}$ contains 3 neurons per stack: one which encodes the content of the stack, one which reads the top element of the stack, and one which stores the answer of the emptyness test of the stack. Moreover, $\mathcal{N}$ contains a pool of neurons which encodes the possible computational states of $\mathcal{S}$ and another pool of neurons which implements the transition function of $\mathcal{S}$. Accordingly, given any computational state, input bit, and contents of the stacks, the

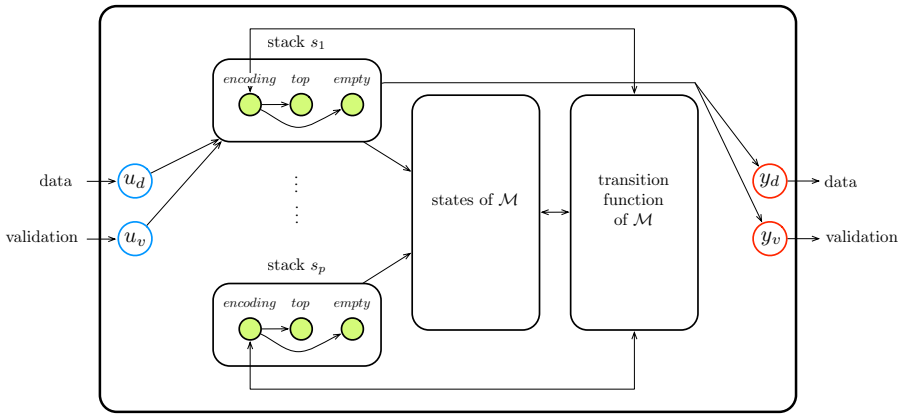


Figure 12 – A St-RNN[Q] $\mathcal{N}$ which simulates a p-stack machine $\mathcal{M}$.

network $\mathcal{N}$ computes a next computational state and updates the contents of the stacks in accordance with the transition function of $\mathcal{S}$. In this way, the network $\mathcal{N}$ simulates correctly the behavior of the p-stack machine $\mathcal{S}$. Furthermore, it can be remarked the the simulation is performed in real time. The network $\mathcal{N}$ is illustrated in Figure 12.

5.4 Real-Weighted or Analog Neural Networks

Static real-weighted recurrent neural networks – or so-called 'analog' RNNs – are actually strictly more powerful than their rational counterparts, and hence also than Turing machines [151, 154]. More precisely, in exponential time of computation, the analog neural networks are capable of unbounded capabilities. When restricted to polynomial time of computation, the networks are computationally equivalent to Turing machines with polynomially bounded advice (TM/poly(A))[3]. Since TM/poly(A)s are strictly more powerful than TMs (i.e., they can decide strictly more languages), it follows that the analog RNNs are capable of extra-recursive computational capabilities from polynomial time of computation already. In this precise sense, they are *super-Turing*.

More precisely, let us defines *static real-weighted – or analog – recurrent neural networks*, denoted by St-RNN[$\mathbb{R}$]s as formal neural networks as defined in previous Section 5.3, with a dynamics governed by Equations (5.2) and (5.3), and provided with static real weights and bias.[4] The result can thus be expressed as follows [151, 154].

Theorem 6. *St-RNN[$\mathbb{R}$]s are super-Turing. More precisely:*

(a) A language L is decidable in polynomial time by some St-RNN[$\mathbb{R}$] iff L is decidable in polynomial time by some TM/poly(A), i.e., iff $L \in$ **P/poly**.

[3] We recall that the set of languages decidable in polynomial time by some TM/poly(A) corresponds to the complexity class **P/poly**.

[4] We recall that a weight $w(t)$ is static and real if $w(t) = C \in \mathbb{R}$, for all $t \geq 0$.

(b) Any language L can be decided in exponential time by some St-RNN[$\mathbb{R}$].

We provide a sketch of the proof of this result. First, let $L \in$ **P/poly**. By some alternative characterization of **P/poly**, there exists a polynomial size circuits family[5] $\mathcal{C} = \{C_n : n \geq 0\}$ such that each circuit C_n decides the sub-language of all words of length n of L, i.e., $L \cap \{0,1\}^n$ [16].

We now consider some recursive encoding of the circuit family $\mathcal{C}$ into a real number $r(\mathcal{C})$. More precisely, first, each circuit C_n of $\mathcal{C}$ is encoded by some finite word $w_{C_n} \in \{0,2,4,6\}^*$, and the whole family $\mathcal{C}$ is encoded by the infinite word $w_{\mathcal{C}} = 8w_{C_0}8w_{C_1}8w_{C_2}8\cdots \in \{0,2,4,6,8\}^\omega$. Afterwards, the infinite word $w_{\mathcal{C}}$ is encoded by the real number $r(\mathcal{C}) = \sum_{i=0}^{\infty} \frac{w_{\mathcal{C}}(i)}{9^i}$.

According to this encoding, one can build some St-RNN[$\mathbb{R}$] $\mathcal{N}$ which contains the real number $r(\mathcal{C})$ as a synaptic weight, and which, given some input u of length n, decodes the circuit C_n of the family $\mathcal{C}$ from $r(\mathcal{C})$, simulates it, and outputs the result in polynomial time. Since the circuits family $\mathcal{C}$ decides L, so does $\mathcal{N}$ in polynomial time. Therefore, $L(\mathcal{N}) = L$. The network $\mathcal{N}$ is illustrated in Figure 13.

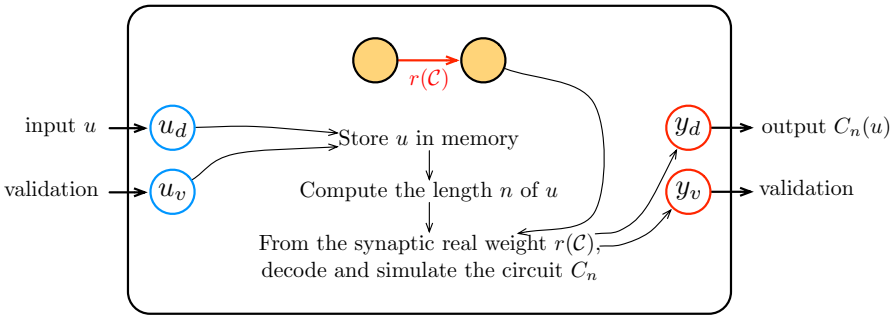


Figure 13 – A St-RNN[$\mathbb{R}$] $\mathcal{N}$ which simulates a polynomial size circuits family $\mathcal{C} = \{C_n : n \geq 0\}$.

The converse implication is more subtle and relies on an important property of recurrent neural networks. Let L be decidable in polynomial time p by some St-RNN[$\mathbb{R}$] $\mathcal{N}$. Then, by some technical lemma, there exists a family of St-RNN[$\mathbb{Q}$]s $\{\mathcal{N}_{p(n)} : n \geq 0\}$ which suitably approximates $\mathcal{N}$, in the sense that each network $\mathcal{N}_{p(n)}$ satisfies the following properties:

- the synaptic weights of $\mathcal{N}_{p(n)}$ are the same as those of $\mathcal{N}$ but truncated after $K \cdot p(n)$ bits, for some constant K.
- if one truncates the successive activation values of every neuron of $\mathcal{N}_{p(n)}$ after $K \cdot p(n)$ bits, then $\mathcal{N}_{p(n)}$ still provide the same outputs as $\mathcal{N}$ up to time $p(n)$.

Such a network $\mathcal{N}_{p(n)}$ is illustrated in Figure 14.

According to these considerations, one can build a TM/poly(A) $\mathcal{M}$ which simulates the behavior of the St-RNN[$\mathbb{R}$] $\mathcal{N}$ in polynomial time. The advice function

[5] For a precise definition of a circuits family, see [16].

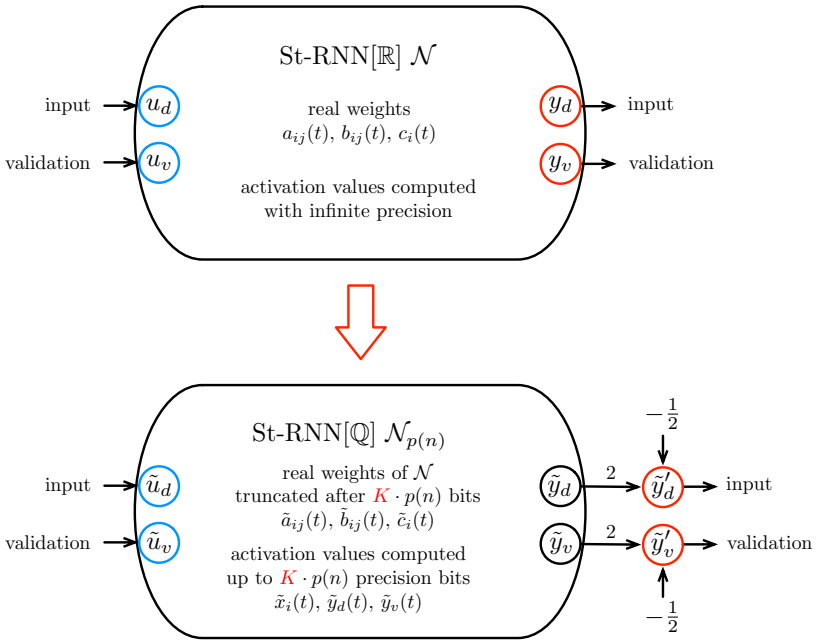


Figure 14 – Relationship between the St-RNN[ℝ] $\mathcal{N}$ and a St-RNN[ℚ] $\mathcal{N}_{p(n)}$ of a family $\{\mathcal{N}_{p(n)} : n \geq 0\}$ which suitably approximates $\mathcal{N}$. The networks $\mathcal{N}$ and $\mathcal{N}_{p(n)}$ provide the same outputs up to time $p(n)$.

is given by $\alpha(n) = \ulcorner \mathcal{N}_{p(n)} \urcorner$ for each $n \geq 0$, where $\ulcorner \mathcal{N}_{p(n)} \urcorner$ is some suitable polynomially bounded binary encoding of $\mathcal{N}_{p(n)}$. The machine is designed such that, on every input u of length n, it first calls the advice value $\ulcorner \mathcal{N}_{p(n)} \urcorner$, decodes the description of $\mathcal{N}_{p(n)}$ from it, and then simulates the behaviour of $\mathcal{N}_{p(n)}$ on u in polynomial time. In this way, the machine $\mathcal{M}$ provides the same output as $\mathcal{N}_{p(n)}$, which itself provides the same output as $\mathcal{N}$, for all inputs of length n. Since $\mathcal{N}$ and $\mathcal{M}$ provide the same answer on every input u, they decide the same language, i.e., $L(\mathcal{M}) = L$. Therefore, $L \in$ **P/poly**. The TM/poly(A) $\mathcal{M}$ is illustrated in Figure 15.

Finally, Theorems 5 and 6 show that the introduction of real synaptic weights in a standard first-order neural model drastically increases the computational power of the networks from Turing to super-Turing capabilities.

5.5 Evolving Neural Networks

5.5.1 The Model

We now characterize the computational capabilities of various kinds of *evolving recurrent neural networks*, i.e., networks where the synaptic weights can update at each discrete time step.

More precisely, we consider once again formal recurrent neural networks, as defined in previous Section 5.3, and with a dynamics governed by Equations (5.2) and (5.3). In this case, six models of RNNs can be considered according to the

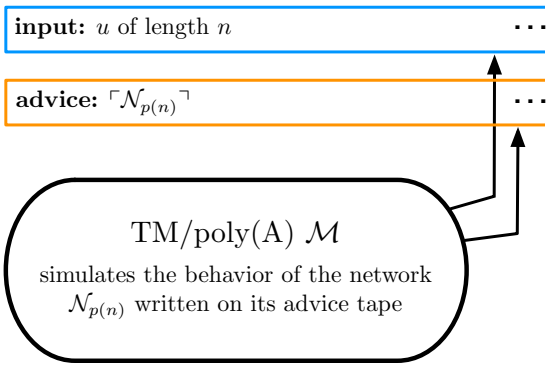


Figure 15 – A TM/poly(A) $\mathcal{M}$ which simulates the behavior of a St-RNN[$\mathbb{R}$] $\mathcal{N}$ in polynomial time.

nature of their synaptic weights (the two first ones have already been considered).

1. the *static rational RNNs* (St-RNN[$\mathbb{Q}$]s) refer to the class of all RNNs whose every weights are static and modelled by rational values.
2. the *static real (or analog) RNNs* (St-RNN[$\mathbb{R}$]s) refer to the class of all RNNs whose every weights are static and modelled by real values.
3. the *bi-valued evolving rational RNNs* (Ev_2-RNN[$\mathbb{Q}$]s) refer to the class of all RNNs whose every evolving weights are bi-valued and every static weights are rational.
4. the *bi-valued evolving real RNNs* (Ev_2-RNN[$\mathbb{R}$]s) refer to the class of all RNNs whose every evolving weights are bi-valued and every static weights are real.
5. the *(general) evolving rational RNNs* (Ev-RNN[$\mathbb{Q}$]s) refer to the class of all RNNs whose every evolving and static weights are rational.
6. the *(general) evolving real RNNs* (Ev-RNN[$\mathbb{R}$]s) refer to the class of all RNNs whose every evolving and static weights are real.

Since rational numbers are included in real numbers and since static weights are particular evolving weights that remain constant over time, the following strict inclusions hold by definition:

$$\begin{array}{ccccc} \text{St-RNN}[\mathbb{Q}]\text{s} & \subsetneq & \text{Ev}_2\text{-RNN}[\mathbb{Q}]\text{s} & \subsetneq & \text{Ev-RNN}[\mathbb{Q}]\text{s} \\ \upharpoonright\!\cap & & \upharpoonright\!\cap & & \upharpoonright\!\cap \\ \text{St-RNN}[\mathbb{R}]\text{s} & \subsetneq & \text{Ev}_2\text{-RNN}[\mathbb{R}]\text{s} & \subsetneq & \text{Ev-RNN}[\mathbb{R}]\text{s} \end{array}$$

Note that the various evolving neural models described above can capture important architectural evolving capabilities other than the sole synaptic plasticity. For instance, creation or deterioration of synapses can be modelled by switching the corresponding synaptic weights on or off, respectively, and cell birth and death are modelled by simultaneously switching on or off all the adjacent synaptic weights of a given cell, respectively.

5.5.2 Computational Power

We show that evolving recurrent neural networks are capable of breaking the Turing barrier of computation. We then provide a precise characterization of their computational power.

More precisely, we first show that rational-weighted evolving recurrent neural networks provided with only bi-valued evolving capabilities are computationally equivalent to static analog neural networks, and therefore, are super-Turing (Theorem 12). We next show that evolving recurrent neural networks remain super-Turing equivalent irrespective of whether their synaptic weights are modelled by rational or real numbers, and moreover, irrespective of whether their patterns of evolvability are restricted to bi-valued updates or expressed by any other more general form of updating (Theorems 13 and 14). Consequently, the four models of Ev_2-RNN[$\mathbb{Q}$]s, Ev-RNN[$\mathbb{Q}$]s), Ev_2-RNN[$\mathbb{R}$]s, and Ev-RNN[$\mathbb{R}$]s are all super-Turing computationally equivalent (Corollary 15). These considerations are summarized in Table 1.

	Static	Bi-valued Evolving	General Evolving
$\mathbb{Q}$	St-RNN[$\mathbb{Q}$]s	Ev_2-RNN[$\mathbb{Q}$]s	Ev-RNN[$\mathbb{Q}$]s
	Turing	super-Turing	super-Turing
$\mathbb{R}$	St-RNN[$\mathbb{R}$]s	Ev_2-RNN[$\mathbb{R}$]s	Ev-RNN[$\mathbb{R}$]s
	super-Turing	super-Turing	super-Turing

Table 1 – Computational power of static and evolving neural networks according to the nature of their synaptic weights and patterns of evolvability.

We now turn to the proofs of the results. These rely on a key technical lemma which is interesting on its own. It is a generalization of the so-called "linear-precision suffices lemma" [154, Lemma 4.1]. Intuitively, the result states that for every Ev-RNN[$\mathbb{R}$] $\mathcal{N}$ deciding some language L in time f, there exists a family of Ev-RNN[$\mathbb{Q}$]s $\{\mathcal{N}_{f(n)} : n > 0\}$ such that each network $\mathcal{N}_{f(n)}$ can compute precisely like $\mathcal{N}$ up to time step $f(n)$ by using only about $f(n)$ precision bits to describe its weights and activation values at every time steps. In other words, every Ev-RNN[$\mathbb{R}$] $\mathcal{N}$ can be approximated by some Ev-RNN[$\mathbb{Q}$] $\mathcal{N}_{f(n)}$ – in the sense that $\mathcal{N}_{f(n)}$ computes precisely like $\mathcal{N}$ up to time step $f(n)$ – where $\mathcal{N}_{f(n)}$ only requires about $f(n)$ precision bits for each instantaneous description.

Formally, consider some Ev-RNN[$\mathbb{R}$] $\mathcal{N}$ given by its input neurons u_d and u_v, its output neurons y_d and y_v, the sequence of its internal neurons $(x_i)_{i=1}^{N-2}$, and the sequence of its evolving weights $(a_{ij}(t))_{t\geq 0}$, $(b_{ij}(t))_{t\geq 0}$, $(c_i(t))_{t\geq 0}$. Consider also some non-decreasing function $f : \mathbb{N}^* \to \mathbb{N}^*$. An *$f$-truncated family over $\mathcal{N}$* consists of a family of Ev-RNN[$\mathbb{Q}$]s $\{\mathcal{N}_{f(n)} : n > 0\}$ where each network $\mathcal{N}_{f(n)}$ is described as follows (cf. Figure 16):

- the non-outputting part of network $\mathcal{N}_{f(n)}$ contains the same number of cells as $\mathcal{N}$, denoted by $\tilde{u}_d$, $\tilde{u}_v$, $\tilde{y}_d$, $\tilde{y}_v$, and $(\tilde{x}_i)_{i=1}^{N-2}$, and the same connectivity patterns between those cells as $\mathcal{N}$;

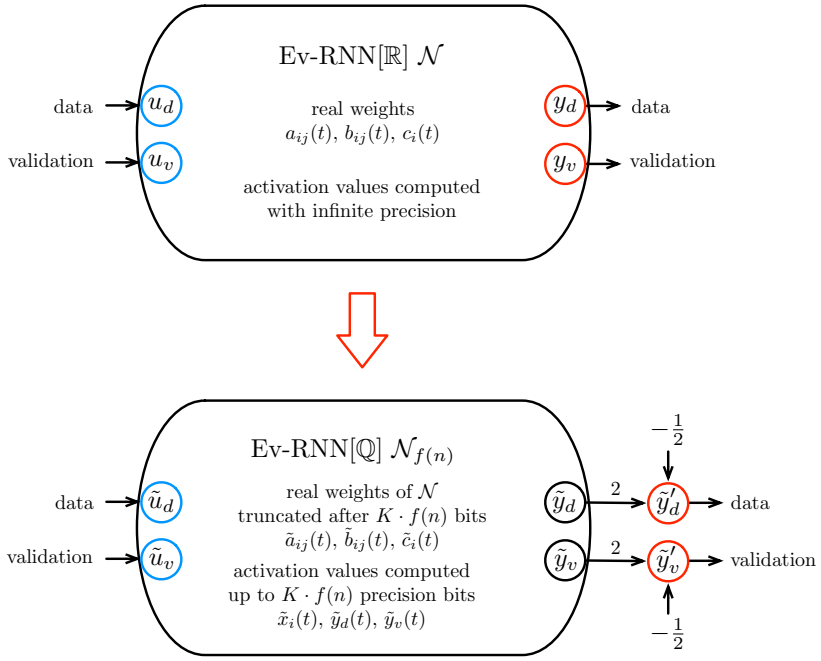


Figure 16 – Relationship between an Ev-RNN[ℝ] $\mathcal{N}$ and the Ev-RNN[ℚ] $\mathcal{N}_{f(n)}$ of an f-truncated family over $\mathcal{N}$.

- the outputting part of $\mathcal{N}_{f(n)}$ consists of two additional cells $\tilde{y}'_d$ and $\tilde{y}'_v$ playing the role of output processors and related to $\tilde{y}_d$ and $\tilde{y}_v$ by the relation $\tilde{y}'_d(t+1) = \sigma(2 \cdot \tilde{y}_d(t) - \frac{1}{2})$ and $\tilde{y}'_v(t+1) = \sigma(2 \cdot \tilde{y}_v(t) - \frac{1}{2})$;
- the dynamics of $\mathcal{N}_{f(n)}$ is given as follows: at each time step, the weights of $\mathcal{N}_{f(n)}$, denoted by $\tilde{a}_{ij}(t)$, $\tilde{b}_{ij}(t)$, $\tilde{c}_i(t)$, correspond to the weights $a_{ij}(t)$, $b_{ij}(t)$, $c_i(t)$ of $\mathcal{N}$ truncated after $K \cdot f(n)$ bits, for some constant K independent of n, and the activation values of all non-outputting processors of $\mathcal{N}_{f(n)}$ at time t, denoted by $\tilde{x}_i(t)$, $\tilde{y}_d(t)$, $\tilde{y}_v(t)$, are computed only up to $K \cdot f(n)$ precision bits, for the same some constant K independent of n.

The relationship between a Ev-RNN[ℝ] $\mathcal{N}$ and the Ev-RNN[ℚ] $\mathcal{N}_{f(n)}$ of an f-truncated family over $\mathcal{N}$ is illustrated in Figure 16. As explained in more detail in the forthcoming proof of Lemma 7, the networks $\mathcal{N}_{f(n)}$ are designed in such a way that the output processors y_d, y_v and $\tilde{y}'_d, \tilde{y}'_v$ of the respective networks $\mathcal{N}$ and $\mathcal{N}_{f(n)}$ would generate the very same binary output values as long as the activation values between y_d, y_v and $\tilde{y}_d, \tilde{y}_v$ are not too distant (this distance being arbitrarily chosen as $\frac{1}{4}$ in our case).

The following result shows that any Ev-RNN[ℝ] can be perfectly simulated by a family of Ev-RNN[ℚ]s in a precise sense.

Lemma 7. *Let $\mathcal{N}$ be some Ev-RNN[ℝ] and $f : \mathbb{N}^* \to \mathbb{N}^*$ be some non-decreasing function. Then there exists an f-truncated family $\{\mathcal{N}_{f(n)} : n > 0\}$ of Ev-RNN[ℚ]s over $\mathcal{N}$ such that, for every input u and every $n > 0$, the output processors of $\mathcal{N}$ and $\mathcal{N}_{f(n)}$ satisfy $y_d(t) = \tilde{y}'_d(t+1)$ and $y_v(t) = \tilde{y}'_v(t+1)$ for all time steps $t \leq f(n)$.*

Proof. By definition of an f-truncated family $\{\mathcal{N}_{f(n)} : n > 0\}$ over $\mathcal{N}$, for each $n > 0$, the dynamics of the non-output processors of $\mathcal{N}_{f(n)}$ is defined by $\tilde{x}_i(0) = 0$ and

$$\tilde{x}_i(t+1) = \left[\sigma\Big(\sum_{j=1}^{N} \big[a_{ij}(t)\big]_{K\cdot f(n)} \cdot \tilde{x}_j(t) + \sum_{j=1}^{M} \big[b_{ij}(t)\big]_{K\cdot f(n)} \cdot \tilde{u}_j(t) \;\; + [c_i(t)]_{K\cdot f(n)}\Big)\right]_{K\cdot f(n)} \tag{5.4}$$

for $i = 1, \ldots, N$, where $[\alpha]_{K\cdot f(n)}$ denotes the value of α truncated after $K \cdot f(n)$ bits for some constant K (independent of n), and the dynamics of the two output processors $\tilde{y}'_d$ and $\tilde{y}'_d$ is given by

$$\begin{aligned} \tilde{y}'_d(t+1) &= \sigma(2 \cdot \tilde{y}_d(t) - \frac{1}{2}) \\ \tilde{y}'_v(t+1) &= \sigma(2 \cdot \tilde{y}_v(t) - \frac{1}{2}). \end{aligned}$$

In order to prove the existence of an f-truncated family $\{\mathcal{N}_{f(n)} : n > 0\}$ over $\mathcal{N}$ with the required properties, we need to prove the existence of a constant K such that, for every $n > 0$ and on every input $u \in \{0,1\}^+$, the Ev-RNN[Q] $\mathcal{N}_{f(n)}$ and the Ev-RNN[$\mathbb{R}$] $\mathcal{N}$ whose dynamics are respectively governed by Equations (5.4) as well as (5.2) and (5.3) actually satisfy $y_d(t) = \tilde{y}'_d(t+1)$ and $y_v(t) = \tilde{y}'_v(t+1)$, for all time steps $t \leq f(n)$. Given some $n > 0$, some input $u \in \{0,1\}^+$, and some time step $t \geq 0$, let $u_d(t)$, $u_v(t)$, $(x_i(t))_{i=1}^{N-2}$, $x_{N-1}(t) = y_d(t)$, $x_N(t) = y_v(t)$ be the activation values of $\mathcal{N}$ at time t, and let $a_{ij}(t)$, $b_{ij}(t)$, $c_i(t)$ be the weights of $\mathcal{N}$ at time t, when working on input u; similarly, let $\tilde{u}_d(t)$, $\tilde{u}_v(t)$, $(\tilde{x}_i(t))_{i=1}^{N-2}$, $\tilde{x}_{N-1}(t) = \tilde{y}_d(t)$, $\tilde{x}_N(t) = \tilde{y}_v(t)$, $\tilde{y}'_d(t)$, $\tilde{y}'_v(t)$ denote the activation values of network $\mathcal{N}_{f(n)}$ at time t, and let $\tilde{a}_{ij}(t) = \big[a_{ij}(t)\big]_{K\cdot f(n)}$, $\tilde{b}_{ij}(t) = \big[b_{ij}(t)\big]_{K\cdot f(n)}$, $\tilde{c}_i(t) = [c_i(t)]_{K\cdot f(n)}$ denote the weights of network $\mathcal{N}_{f(n)}$ at time t, when working on u. Note that since we consider the same input u, one has $u_d(t) = \tilde{u}_d(t)$ and $u_v(t) = \tilde{u}_v(t)$. Let also $W = \max\{|S|, |S'|\}$, where S and S' are the bounds on the weights of $\mathcal{N}$. Furthermore, let the largest truncation errors of the processors and weight at time t as well as the largest accumulated error at time t be given by:

$$\begin{aligned} \delta_p(u,t) = \max_i \Bigg| &\Big[\sigma\Big(\sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot \tilde{x}_j(t) + \sum_{j=1}^{M} \tilde{b}_{ij}(t) \cdot u_j(t) + \tilde{c}_i(t)\Big)\Big]_{K\cdot f(n)} \\ &- \sigma\Big(\sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot \tilde{x}_j(t) + \sum_{j=1}^{M} \tilde{b}_{ij}(t) \cdot u_j(t) + \tilde{c}_i(t)\Big)\Bigg| \end{aligned}$$

$$\delta_w(t) = \max\big\{\max_{i,j} |\tilde{a}_{ij}(t) - a_{ij}(t)|\,,\ \max_{i.j} |\tilde{b}_{ij}(t) - b_{ij}(t)|\,,\ \max_i |\tilde{c}_i(t) - c_i(t)|\big\}$$

$$\epsilon(t) = \max_i |\tilde{x}_i(t) - x_i(t)|$$

Now, let δ_p be the supremum of all values $\delta_p(u,t)$ over all possible inputs u and time steps t, i.e. $\delta_p = \sup_{u\in\{0,1\}^+, t\geq 0} \delta_p(u,t)$. Since any possible activation value always belongs to $[0,1]$, one has $\delta_p(u,t) \leq \delta_p \leq 1$. Let also δ_w be the supremum of all values $\delta_w(t)$ over all possible time steps (note that $\delta_w(t)$ does not depend on the input u), i.e. $\delta_w = \sup_{t\geq 0} \delta_w(t)$. Since every possible weight belongs by definition

to $[s, s']$, the weight's largest truncation errors cannot exceed $W = \max\{|S|, |S'|\}$, and one thus has $\delta_w(t) \leq \delta_w \leq W$.

According to these definitions, using the global Lipschitz property $|\sigma(x) - \sigma(y)| \leq |x - y|$ and the fact that $u_j(t) = \tilde{u}_j(t)$, one has

$$
\begin{aligned}
\epsilon(t+1) = & \max_i \quad |\tilde{x}_i(t+1) - x_i(t+1)| \\
= & \max_i \quad \Big| \Big[\sigma\Big(\sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot \tilde{x}_j(t) + \sum_{j=1}^{M} \tilde{b}_{ij}(t) \cdot u_j(t) + \tilde{c}_i(t)\Big)\Big]_{K \cdot f(n)} - \\
& \quad \sigma\Big(\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)\Big)\Big| \\
\leq & \max_i \quad \Big| \sigma\Big(\sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot \tilde{x}_j(t) + \sum_{j=1}^{M} \tilde{b}_{ij}(t) \cdot u_j(t) + \tilde{c}_i(t)\Big) - \\
& \quad \sigma\Big(\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)\Big)\Big| + \\
& \quad \delta_p(u,t) \\
\leq & \max_i \quad \Big| \Big(\sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot \tilde{x}_j(t) + \sum_{j=1}^{M} \tilde{b}_{ij}(t) \cdot u_j(t) + \tilde{c}_i(t)\Big) - \\
& \quad \Big(\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)\Big)\Big| + \\
& \quad \delta_p(u,t) \\
\leq & \max_i \quad \Big| \sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot \tilde{x}_j(t) - \sum_{j=1}^{N} a_{ij}(t) \cdot x_j(t)\Big| + \\
& \quad \Big| \sum_{j=1}^{M} \tilde{b}_{ij}(t) \cdot u_j(t) - \sum_{j=1}^{M} b_{ij}(t) \cdot u_j(t)\Big| + \\
& \quad \Big|\tilde{c}_i(t) - c_i(t)\Big| + \delta_p(u,t) \\
\leq & \max_i \quad \Big| \sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot \tilde{x}_j(t) - \sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot x_j(t)\Big| + \\
& \quad \Big| \sum_{j=1}^{N} \tilde{a}_{ij}(t) \cdot x_j(t) - \sum_{j=1}^{N} a_{ij}(t) \cdot x_j(t)\Big| + \\
& \quad \Big| \sum_{j=1}^{M} \tilde{b}_{ij}(t) \cdot u_j(t) - \sum_{j=1}^{M} b_{ij}(t) \cdot u_j(t)\Big| + \\
& \quad \Big|\tilde{c}_i(t) - c_i(t)\Big| + \delta_p(u,t) \\
\leq & \quad N \cdot (W + \delta_w(t)) \cdot \epsilon(t) + (N + M + 1) \cdot \delta_w(t) + \delta_p(u,t) \\
\leq & \quad N \cdot 2 \cdot W \cdot \epsilon(t) + (N + M + 1) \cdot \delta_w(t) + \delta_p(u,t) \\
\leq & \quad N \cdot 2 \cdot W \cdot \epsilon(t) + (N + M + 1) \cdot \delta_w + \delta_p
\end{aligned}
$$

$$= \quad K_1 \cdot \epsilon(t) + K_2 \cdot \delta_w + \delta_p$$

where $K_1 = 2 \cdot N \cdot W$ and $K_2 = N + M + 1$ are constants. Using the fact that $\epsilon(0) = \max_i |\tilde{x}_i(0) - x_i(0)| = 0$ and the geometric series formula, one has

$$\epsilon(t+1) \leq \sum_{i=0}^{t} K_1^i \cdot (K_2 \cdot \delta_w + \delta_p) \leq K_1^{t+1} \cdot (K_2 \cdot \delta_w + \delta_p) \tag{5.5}$$

Now, we want to show the existence of a constant K such that the binary output values $y_d(t), y_v(t)$ and $\tilde{y}'_d(t+1), \tilde{y}'_v(t+1)$ of the respective network $\mathcal{N}$ and truncated network $\mathcal{N}_{f(n)}$ satisfy the relations $y_d(t) = \tilde{y}'_d(t+1)$ and $y_v(t) = \tilde{y}'_v(t+1)$, for all $t \leq f(n)$. In other words, according to the dynamics of $\tilde{y}'_d$ and $\tilde{y}'_v$, one must have for all $t \leq f(n)$:

$$\begin{aligned}
&\text{if } y_d(t) = 0 \text{ then } \sigma\left(2 \cdot \tilde{y}_d(t) - \frac{1}{2}\right) = 0 \text{ i.e. } 2 \cdot \tilde{y}_d(t) - \frac{1}{2} \leq 0 \\
&\text{if } y_d(t) = 1 \text{ then } \sigma\left(2 \cdot \tilde{y}_d(t) - \frac{1}{2}\right) = 1 \text{ i.e. } 2 \cdot \tilde{y}_d(t) - \frac{1}{2} \geq 1 \\
&\text{if } y_v(t) = 0 \text{ then } \sigma\left(2 \cdot \tilde{y}_v(t) - \frac{1}{2}\right) = 0 \text{ i.e. } 2 \cdot \tilde{y}_v(t) - \frac{1}{2} \leq 0 \\
&\text{if } y_v(t) = 1 \text{ then } \sigma\left(2 \cdot \tilde{y}_v(t) - \frac{1}{2}\right) = 1 \text{ i.e. } 2 \cdot \tilde{y}_v(t) - \frac{1}{2} \geq 1
\end{aligned}$$

for all $t \leq f(n)$. Note that the above relations hold whenever $|\tilde{y}_d(t) - y_d(t)| \leq \frac{1}{4}$ and $|\tilde{y}_v(t) - y_v(t)| \leq \frac{1}{4}$ for all $0 \leq t \leq f(n)$, and hence in particular whenever $\epsilon(t) \leq \frac{1}{4}$ for all $t \leq f(n)$ (since y_d and y_v belong to the x_i's). This latter inequality is satisfied for $t = 0$ (since $\epsilon(0) = 0 \leq \frac{1}{4}$), and according to Inequality (5.5), it also holds for all $0 < t \leq f(n)$ if

$$K_1^t \cdot (K_2 \cdot \delta_w + \delta_p) \leq \frac{1}{4} \text{ for all } 0 < t \leq f(n).$$

Now, note that the previous relations hold if δ_w and δ_p are both bounded by all values $\frac{1}{5}(K_1 \cdot K_2)^{-t}$, for all $0 < t \leq f(n)$ (for the case $t = 1$, use the fact that $K_2 \geq 5$, and for the cases $t > 1$, use the fact that $K_2 + 1 \leq K_2^t$). Since $\frac{1}{5}(K_1 \cdot K_2)^{-f(n)} \leq \frac{1}{5}(K_1 \cdot K_2)^{-t}$ for all $t \leq f(n)$, the above relations are also satisfied if δ_w and δ_p are both bounded by $\frac{1}{5}(K_1 \cdot K_2)^{-f(n)}$. Moreover, we recall that if the truncations in δ_w and δ_p occur at $K \cdot f(n)$ bits after the decimal point, then δ_w and δ_p are bounded by $2^{-K \cdot f(n)}$. Hence, in order to have δ_w and δ_p bounded by $\frac{1}{5}(K_1 \cdot K_2)^{-f(n)}$ as requested, it suffices to have $2^{-K \cdot f(n)} \leq \frac{1}{5}(K_1 \cdot K_2)^{-f(n)}$, i.e., to have $K \geq \log(\frac{1}{5}(K_1 \cdot K_2))$. Therefore, by taking $K = \lceil \log(\frac{1}{5}(K_1 \cdot K_2)) \rceil$ (where $\lceil x \rceil$ denotes the least integer above x), the dynamics given by Equation (5.4) ensures that the output binary values $y_d(t), y_v(t)$ as well as $\tilde{y}'_d(t+1), \tilde{y}'_v(t+1)$ produced by the respective networks $\mathcal{N}$ and $\mathcal{N}_{f(n)}$ will be the very same for all time steps $t \leq f(n)$. This concludes the proof. □

The following Propositions 8 and 9 constitute the core of the main results.

Proposition 8. *Let $L \subseteq \{0,1\}^+$ be some language.*

(i) There exists some Ev$_2$-RNN[$\mathbb{Q}$] that decides L in exponential time.

(ii) There exists some Ev-RNN[$\mathbb{Q}$] that decides L in exponential time.

Proof. Note that Point (ii) is a direct consequence of Point (i), since any Ev$_2$-RNN[Q] is a particular Ev-RNN[Q]. We now prove Point (i). The main idea of the proof is illustrated in Figure 17.

We prove Point (i). First of all, let $w_1, w_2, w_3, \ldots$ denote the infinite lexicographical enumeration of all words of $\{0,1\}^+$ (i.e. $w_1 = 0$, $w_2 = 1$, $w_3 = 00$, $w_4 = 01$, $w_5 = 10$, $w_6 = 11$, $w_7 = 000$, etc.), and for every $i > 0$, let ε_i be the *L-characteristic bit* $\chi_L(w_i)$ of w_i, i.e. $\varepsilon_i = 1$ iff $w_i \in L$. Now, let w be the binary infinite word defined as the succession of all w_i's and ε_i's separated by 0's, i.e.

$$w = w_1 0 \varepsilon_1 0 w_2 0 \varepsilon_2 0 w_3 0 \varepsilon_3 0 w_4 0 \varepsilon_4 0 \cdots .$$

In words, w represents a description of all successive binary words followed by the information of whether each of these words belongs to L or not. Besides, consider also the binary infinite word z which has the same structure as w except that every sub-word w_i is replaced by a block of 1's of the same length and every bit ε_i is replaced by a 1, i.e.

$$z = 101010101101011010 \cdots .$$

The idea is that the infinite word z acts as a validation line for the infinite word w, i.e., the positions of the active bits of z correspond to those of the data bits of w. The superposition bit by bit of the words w and z is as follows:

$$\begin{array}{cccccccccccccccc} w_1 & 0 & \varepsilon_1 & 0 & w_2 & 0 & \varepsilon_2 & 0 & w_3 & 0 & \varepsilon_3 & 0 & w_4 & 0 & \varepsilon_4 & 0 & \cdots \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 11 & 0 & 1 & 0 & 11 & 0 & 1 & 0 & \cdots \end{array}$$

We provide the description of a Ev$_2$-RNN[Q] $\mathcal{N}_L$ that decides L in exponential time. The network $\mathcal{N}_L$ actually consists of one evolving and one static rational subnetwork connected together.

The evolving rational-weighted part of $\mathcal{N}_L$ is made up of two designated processors x_p and $x_{p'}$. Both neurons x_p and $x_{p'}$ receive as incoming synaptic connections a background activity of changing intensity $c_p(t)$ and $c_{p'}(t)$, respectively. The synaptic weight $c_p(t)$ takes as values the successive bits composing the infinite word w. The synaptic weight $c_{p'}(t)$ takes as values the successive bits composing the infinite word z. At each time step, $c_p(t)$ and $c_{p'}(t)$ switch from one bit to the next. In this way, the synaptic weights $c_p(t)$ and $c_{p'}(t)$ evolve among only two possible values, namely 0 or 1. In words, the neurons x_p and $x_{p'}$ respectively receive as background activities the two binary infinite words w and z in parallel.

The static rational-weighted part of $\mathcal{N}_L$ is designed in order to perform the recursive neural procedure described by Algorithm 1 below. Algorithm 1 receives some finite binary input u bit by bit, uses the information provided by the evolving neurons x_p and $x_{p'}$, and eventually decides whether u belongs to L or not.

Algorithm 1 consists of two subroutines performed in parallel until some return instruction is eventually reached. It involves 7 designated neurons x_u, x_p, $x_{p'}$, x_w,

$x_{w'}$, x_z, $x_{z'}$. Concerning instructions 4, 5 and 12, the way to store successive incoming bits into some designated neuron is described in details in [155]. Intuitively, the activation value of a neuron can be employed to encode the content a binary stack, and every new incoming bit can be pushed into the stack in constant time [155]. These tasks can be achieved by some simple static rational-weighted RNNs [155]. In order to be able to store binary words of any possible finite length, one needs to dispose of an unbounded memory. This is achieved via the possibility to dispose of an arbitrary precision for the rational activation values of the neurons [155]. For instructions 7 and 14, the implementation of a counter by some static rational RNN is described in details in [155]. The counter is implemented as a unary stack. Incrementing or decrementing the counter is achieved by pushing or popping an element to and from the stack. The content of the unary stack is encoded by the activation value of a neuron [155]. In instructions 15 and 16, the two copies are achieved by simply triggering two synaptic connections of intensities 1 from x_w to $x_{w'}$, and from x_z to $x_{z'}$, respectively. Instructions 17 to 23 are written in a high-level language, but it is clear that they can be performed by some three-tape Turing machine, where the words u, w', and z' encoded in neurons x_u, $x_{w'}$, and $x_{z'}$ are written on each tape at the beginning of the computation. According to the real-time computational equivalence between Turing machines and static rational-weighted RNNs [155], this instruction block can also be simulated by some static rational RNNs with the words u, w' and z' encoded as rational activation values of the three designated neurons x_u, $x_{w'}$ and $x_{z'}$ at the beginning of the computation. The possibility to encode and decode any finite binary word into and from the activation value of some neuron is described in details in [155].

Besides, note that every time instructions 14, 15, 16 are re-executed (via instruction 22), the activation values of $x_{w'}$ and $x_{z'}$ will represent the encodings of two words w' and z' which strictly extend those two involved in the previous execution of these instructions. Now, since $(w_i)_{i>0}$ is an enumeration of $\{0,1\}^+$, there exists some $k > 0$ such that $u = w_k$. By the previous argument, there will necessarily be some execution of instructions 14, 15, 16 involving a word w' of the form $w' = w_1 0 \varepsilon_1 0 w_2 0 \varepsilon_2 0 \cdots w_n 0 \varepsilon_n 0$, where $n \geq k$. In this case, the word u matches one of the sub-words w_i's of w', meaning that Algorithm 1 will either provide an accepting or a rejecting answer, and therefore necessarily terminate.

Hence, the analysis of each instruction ensures that Algorithm 1 always terminates and can indeed be simulated by some static rational RNN. This RNN represents the static rational-weighted part of $\mathcal{N}_L$.

The Ev_2-RNN[$\mathbb{Q}$] $\mathcal{N}_L$ consists of the bi-valued evolving and the static rational sub-networks described above. According to Algorithm 1, $\mathcal{N}_L$ clearly decides the language L. Moreover, for any input u of length n, the network has to wait for $O(2^n)$ time steps before the binary word u occurs as a sub-word of w. Therefore, the network $\mathcal{N}_L$ decides the language L in exponential time. □

Proposition 9. *Let $L \subseteq \{0,1\}^+$ be some language.*

*(i) L is decidable in polynomial time by some Ev_2-RNN[$\mathbb{Q}$] if and only if $L \in$ **P/poly***

*(ii) L is decidable in polynomial time by some Ev-RNN[$\mathbb{Q}$] if and only if $L \in$ **P/poly***

Algorithm 1 Neural procedure

Require: finite binary word u provided bit by bit

1: **SUBROUTINE 1:**
2: $c \leftarrow 0$
3: **for all** time steps $t \geq 0$ **do**
4: store $x_p(t)$ into neuron x_w
5: store $x_{p'}(t)$ into neuron x_z
6: **if** $x_{p'}(t) = 0$ **then**
7: $c \leftarrow c + 1 \bmod 2$ // c counts modulo 2 the number of 0's occurring at neuron $x_{p'}$
8: **end if**
9: **end for**
10: **SUBROUTINE 2:**
11: **for** $t = 0$ **to** $|u|$ **do**
12: store $u(t)$ into neuron x_u
13: **end for** // the activation value of x_u represents an encoding of u
14: wait for c (of subroutine 1) to switch from 1 to 0
15: copy the current activation value of neuron x_w (of subroutine 1) into neuron $x_{w'}$ // at time step t, the value of $x_{w'}$ represents the encoding of the word w' of the form $w' = w_1 0 \varepsilon_1 0 w_2 0 \varepsilon_2 0 \cdots w_{n(t)} 0 \varepsilon_{n(t)} 0$, for some $n(t) > 0$
16: copy the current activation value of neuron x_z (of subroutine 1) into neuron $x_{z'}$ // at time step t, the value of $x_{z'}$ represents the encoding of the word z' of the form $z' = 1^{|w_1|} 0 \varepsilon_1 0 1^{|w_2|} 0 \varepsilon_2 0 \cdots 1^{|w_{n(t)}|} 0 \varepsilon_{n(t)} 0$, for some $n(t) > 0$
17: **if** $u = w_i$ for some $i = 1, \ldots, n(t)$ and $\varepsilon_i = 1$ **then**
18: **return** ACCEPT // in this case, $u \in L$
19: **else if** $u = w_i$ for some $i = 1, \ldots, n(t)$ and $\varepsilon_i = 0$ **then**
20: **return** REJECT // in this case, $u \notin L$
21: **else**
22: goto instruction 14 // we still don't know whether u belongs to L or not
23: **end if**

The proof is achieved by the two following Lemmas 10 and 11. Note that Lemma 10 concerns Ev_2-RNN[Q]s whereas Lemma 11 concerns Ev-RNN[Q]s.

Lemma 10. *Let $L \subseteq \{0,1\}^+$ be some language. If $L \in$ **P/poly**, then there exists a Ev_2-RNN[Q] that decides L in polynomial time.*

Proof. The present proof resembles the proof of Proposition 8. The main idea of the proof is illustrated in Figure 18.

Since $L \in$ **P/poly**, there exists a TM/poly(A) $\mathcal{M}$ that decides L in polynomial time. Let $\alpha : \mathbb{N}^* \to \{0,1\}^+$ be the polynomially bounded advice function of $\mathcal{M}$. Let w be the binary infinite word defined as the succession of all $\alpha(i)$'s separated by 0's, i.e.

$$w = \alpha(1)0\alpha(2)0\alpha(3)0\alpha(4)0\cdots .$$

Moreover, let z be the binary infinite word which has the same structure as w except

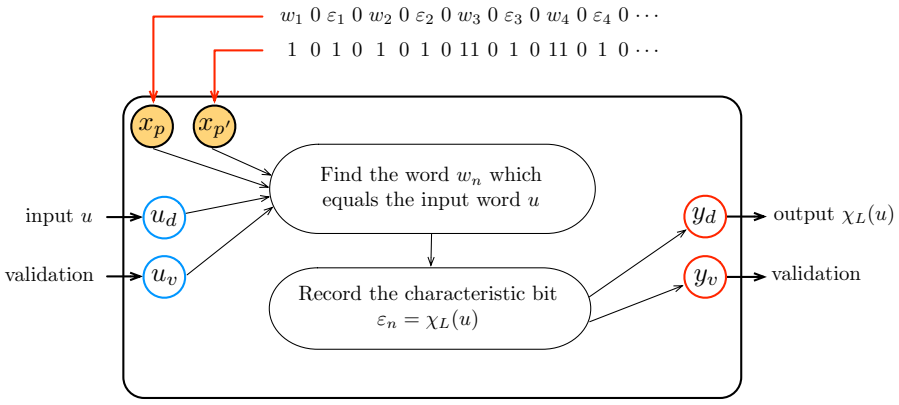


Figure 17 – Illustration of the Ev$_2$-RNN[Q] $\mathcal{N}_L$ described in the proof of Proposition 8.

that every sub-word $\alpha(w_i)$ is replaced by a block of 1's of the same length, i.e.

$$z = 1^{|\alpha(w_1)|}01^{|\alpha(w_2)|}01^{|\alpha(w_3)|}01^{|\alpha(w_4)|}0\cdots.$$

Once again, the idea is that the infinite word z acts as a validation line for the infinite word w, i.e. the active bits of z correspond in their positions to the data bits of w. The superposition bit by bit of the words w and z is as illustrated below:

$$\begin{array}{cccccccccc} \alpha(1) & 0 & \alpha(2) & 0 & \alpha(3) & 0 & \alpha(4) & 0 & \cdots \\ 1^{|\alpha(1)|} & 0 & 1^{|\alpha(2)|} & 0 & 1^{|\alpha(3)|} & 0 & 1^{|\alpha(4)|} & 0 & \cdots \end{array}$$

We now provide the description of a Ev$_2$-RNN[Q] $\mathcal{N}_L$ that decides L in polynomial time. Once again, the network $\mathcal{N}_L$ consists of one evolving and one static rational sub-network connected together.

The evolving rational-weighted part of $\mathcal{N}_L$ is made up of two designated processors x_p and $x_{p'}$. The neurons x_p and $x_{p'}$ receive as incoming synaptic connections background activities of changing intensities $c_p(t)$ and $c_{p'}(t)$, each of which taking as values the successive bits of the infinite words w and z, respectively. At each time step, $c_p(t)$ and $c_{p'}(t)$ switch from one bit to the next. In this way, the synaptic weights $c_p(t)$ and $c_{p'}(t)$ evolve among only two possible values, namely 0 or 1. In words, the neurons x_p and $x_{p'}$ respectively receive as background activities the two binary infinite words w and z in parallel.

The static rational-weighted part of $\mathcal{N}_L$ is designed in order to perform the recursive neural procedure described by Algorithm 2 below. Algorithm 2 receives some finite binary input u bit by bit, uses the information provided by the evolving neurons x_p and $x_{p'}$, and eventually decides in polynomial time whether u belongs to L or not.

Algorithm 2 consists of two subroutines performed in parallel until some final answer is provided in instruction 16. It involves 8 designated neurons x_u, x_p, $x_{p'}$, x_w, $x_{w'}$, x_z, $x_{z'}$, x_α. Concerning instructions 3, 4 and 8, the way to store successive incoming bits into some designated neuron is described in detail in [155]. Instruction 10 uses a counter which implemented as a unary stack. For each letter of u,

a 1 is pushed into the stack. The content of the stack is encoded in the activation value of a neuron [155]. For instruction 11, the counting procedure is implemented as follows: every time some 0 occurs as a background activity of neuron $x_{p'}$, a 1 is popped from the neuron stack, until the stack becomes empty. In instructions 12 and 13, the two copies are achieved by simply triggering two synaptic connections of intensities 1 from x_w to $x_{w'}$, and from x_z to $x_{z'}$, respectively. The block of instructions 14 and 15 is written in a high-level language, but it is clear that it can be performed by some Turing machine, where the words w' encoded in neurons $x_{w'}$ is written on the tape at the beginning of the computation. According to the real-time computational equivalence between Turing machines and static rational-weighted RNNs [155], this sub-procedure can also be simulated by some static rational RNNs with the word w' encoded as the rational activation value of a designated neurons $x_{w'}$ at the beginning of the computation. Concerning instruction 16, the behaviour of a TM/poly(A) $\mathcal{M}$ working on u and with the advice string $\alpha(n)$ already written on its advice tape is clearly recursive (only the call to the advice function is not recursive), and therefore, can be simulated by some static rational-weighted RNN [155].

Hence, the analysis of each instruction ensures that Algorithm 2 can indeed be simulated by some static rational RNN, which represents the static rational-weighted part of $\mathcal{N}_L$.

The Ev$_2$-RNN[$\mathbb{Q}$] $\mathcal{N}_L$ consists of the bi-valued evolving and the static rational sub-networks described above. According to Algorithm 2, the Ev$_2$-RNN[$\mathbb{Q}$] $\mathcal{N}_L$ outputs the same answer as $\mathcal{M}$. Since $\mathcal{M}$ decides the language L, so does $\mathcal{N}_L$. Besides, since the advice is polynomially bounded, it follows that for any input u, the network has to wait for polynomially many time steps before the binary word $\alpha(|u|)$ occurs as a sub-word of w. Moreover, since $\mathcal{M}$ decides L in polynomial time, the simulating task of $\mathcal{M}$ by $\mathcal{N}_L$ is also done in polynomial time in the input size [155]. Consequently, $\mathcal{N}_L$ decides L in polynomial time. □

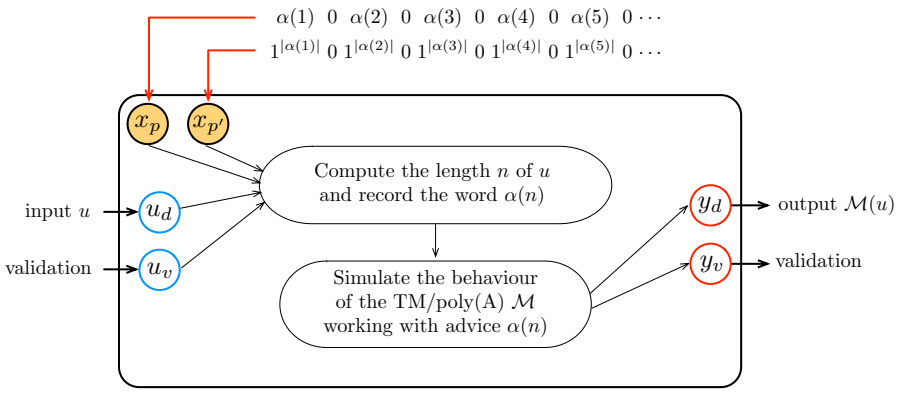


Figure 18 – Illustration of the Ev$_2$-RNN[$\mathbb{Q}$] $\mathcal{N}_L$ described in the proof of Lemma 10.

Lemma 11. *Let $L \subseteq \{0,1\}^+$ be some language. If there exists a Ev-RNN[$\mathbb{Q}$] that decides L in polynomial time, then $L \in$* **P/poly**.

Algorithm 2 Neural procedure

Input: finite binary word u provided bit by bit

1: **SUBROUTINE 1:**
2: **for all** time steps $t \geq 0$ **do**
3: store $x_p(t)$ into neuron x_w
4: store $x_{p'}(t)$ into neuron x_z
5: **end for**
6: **SUBROUTINE 2:**
7: **for** $t = 0$ **to** $|u|$ **do**
8: store $u(t)$ into neuron x_u
9: **end for** // the activation value of x_u represents an encoding of u
10: compute and store the value $|u|$ in a neuron
11: from the current time step, wait that $|u|$ occurrences of 0 have appeared at neuron $x_{p'}$
12: copy the current activation value of neuron x_w of subroutine 1 into neuron $x_{w'}$
13: copy the current activation value of neuron x_z of subroutine 1 into neuron $x_{z'}$ // at that point, the activation values of $x_{w'}$ and $x_{z'}$ represent the encodings of two words w' and z' that are prefixes of w and z respectively. Since one has waited that at least $|u|$ 0's have occurred as a background activity of neuron $x_{p'}$, we are sure that the finite word w' contains the value $\alpha(|u|)$ as subword.
14: decode $\alpha(|u|)$ from the activation value of $x_{w'}$
15: store $\alpha(|u|)$ into neuron x_α
16: simulate the behaviour of the TM/poly(A) $\mathcal{M}$ working on u with $\alpha(n)$ written on its advice tape

Proof. The main idea of the proof is illustrated in Figure 19. Suppose that L is decided by some Ev-RNN[$\mathbb{Q}$] $\mathcal{N}$ in polynomial time p. Since $\mathcal{N}$ is by definition also a Ev-RNN[$\mathbb{R}$], Lemma 7 applies and shows the existence of a p-truncated family of Ev-RNN[$\mathbb{Q}$]s over $\mathcal{N}$. Hence, for every n, there exists a Ev-RNN[$\mathbb{Q}$] $\mathcal{N}_{p(n)}$ such that: firstly, the network $\mathcal{N}_{p(n)}$ has the same processors and connectivity pattern as $\mathcal{N}$; secondly, for every $t \leq p(n)$, each rational synaptic weight of $\mathcal{N}_{p(n)}(t)$ can be represented by some sequence of bits of length at most $C \cdot p(n)$, for some constant C independent of n; thirdly, on every input of length n, if one restricts the activation values of $\mathcal{N}_{p(n)}$ to be all truncated after $C \cdot p(n)$ bits at every time step, then the output processors of $\mathcal{N}$ and $\mathcal{N}_{p(n)}$ at respective time steps t and $t+1$ have the same activation values for all time steps $t \leq p(n)$.

We now prove that L can also be decided in polynomial time by some TM/poly(A) $\mathcal{M}$. First of all, consider the advice function $\alpha : \mathbb{N} \to \{0,1\}^+$ given by $\alpha(i) = \mathit{Encoding}(\langle \mathcal{N}_{p(i)}(t) : 0 \leq t \leq p(i) \rangle)$, where $\mathit{Encoding}(\langle \mathcal{N}_{p(i)}(t) : 0 \leq t \leq p(i) \rangle)$ denotes some suitable recursive encoding of the sequence of successive descriptions of the network $\mathcal{N}_{p(i)}$ up to time step $p(i)$. Note that $\alpha(i)$ consists of the encoding of $p(i)+1$ successive descriptions of the network $\mathcal{N}_{p(i)}$, where each of this description has the same fixed number of processors and synaptic weights, and each synaptic weight being representable by at most $C \cdot p(i)$ bits. Therefore, the length of $\alpha(i)$

belongs to $O(p(i)^2)$, and thus is still polynomial in i.

Now, consider the TM/poly(A) $\mathcal{M}$ that uses α as advice function, and which, on every input u of length n, first calls the advice word $\alpha(n)$, then decodes this sequence in order to simulate the truncated network $\mathcal{N}_{p(n)}$ on input u up to time step $p(n)$ and in such a way that all activation values of $\mathcal{N}_{p(n)}$ are only computed up to $C \cdot p(n)$ bits at every time step. Note that each simulation step of $\mathcal{N}_{p(n)}$ by $\mathcal{M}$ is performed in polynomial time in n, since the decoding of the current configuration of $\mathcal{N}_{p(n)}$ from $\alpha(n)$ is polynomial in n, and the computation and representations of the next activation values of $\mathcal{N}_{p(n)}$ from its current activation values and synaptic weights are also polynomial in n. Consequently, the $p(n)$ simulation steps of $\mathcal{N}_{p(n)}$ by $\mathcal{M}$ are performed in polynomial time in n.

Now, since any u of length n is classified by $\mathcal{N}$ in time $p(n)$, Lemma 7 ensures that u is also classified by $\mathcal{N}_{p(n)}$ in time $p(n)$, and the behaviour of $\mathcal{M}$ ensures that u is also classified by $\mathcal{M}$ in $p(n)$ simulation steps of $\mathcal{N}_{p(n)}$, each of which being polynomial in n. Hence, any word u of length n is classified by the TM/poly(A) $\mathcal{M}$ in polynomial time in n, and the classification answers of $\mathcal{M}$, $\mathcal{N}_{p(n)}$, and $\mathcal{N}$ are the very same. Since $\mathcal{N}$ decides the language L, so does $\mathcal{M}$. Therefore $L \in$ **P/poly**, which concludes the proof. □

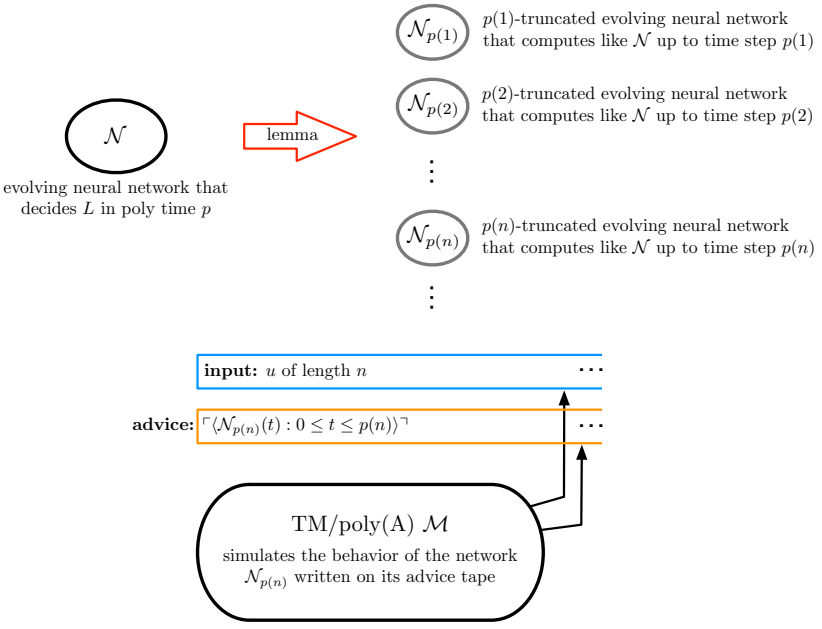


Figure 19 – Illustration of the proof idea of Lemma 11.

Proof of Proposition 9. Concerning Point (i), the backward implication is given by Lemma 10. For the forward implication, suppose that L is decidable by some Ev_2-RNN[Q] $\mathcal{N}$. Then, L is also decidable by some Ev-RNN[Q], namely $\mathcal{N}$ itself. By Lemma 11, $L \in$ **P/poly**.

Concerning Point (ii), the forward implication is given by Lemma 11. For the

backward implication, suppose that $L \in$ **P/poly**. By Lemma 10, L is decidable by some Ev$_2$-RNN[$\mathbb{Q}$] $\mathcal{N}$. Consequently, L is also decidable by some Ev-RNN[$\mathbb{Q}$], namely $\mathcal{N}$ itself. This concludes the proof. □

We can now state the main results of the paper. First, we show that the bi-valued evolving rational recurrent neural networks are super-Turing.

Theorem 12. *Ev$_2$-RNN[$\mathbb{Q}$]s are super-Turing. More precisely:*

*(a) A language L is decidable in polynomial time by some Ev$_2$-RNN[$\mathbb{Q}$] if and only if L is decidable in polynomial time by some TM/poly(A), i.e., iff $L \in$ **P/poly**.*

(b) Any language L can be decided in exponential time by some Ev$_2$-RNN[$\mathbb{Q}$].

Proof. Points (b) and (a) are given by Proposition 8(i) and Proposition 9(i), respectively. □

Moreover, we show that the consideration of more general patterns of evolvability would not increase further the computational capabilities of the neural networks. In other words, the translation from the bi-valued evolving to the general evolving rational context would not provide any additional power to the neural model.

Theorem 13. *Ev-RNN[$\mathbb{Q}$]s are super-Turing equivalent to Ev$_2$-RNN[$\mathbb{Q}$]s in polynomial as well as in exponential time of computation.*

Proof. The exponential time and polynomial time equivalences are given by Proposition 8(ii) and Proposition 9(ii), respectively. □

Now, we prove that the evolving real neural networks are computationally equivalent to the bi-valued evolving rational ones, irrespective of whether their evolving synaptic weight are restricted to bi-valued patterns of evolvability or expressed by any other more general form of updating. Hence, once again, the translation from the evolving rational to the evolving real context would not provide any additional power to the neural model.

Theorem 14. *Both models of Ev$_2$-RNN[$\mathbb{R}$]s and Ev-RNN[$\mathbb{R}$]s are super-Turing equivalent to Ev$_2$-RNN[$\mathbb{Q}$]s in polynomial as well as in exponential time of computation.*

Proof. We first consider the case of the exponential time of computation. Let $L \subseteq \{0,1\}^*$ be some language. Then, by Proposition 8(i), L is decidable in exponential time by some Ev$_2$-RNN[$\mathbb{Q}$] $\mathcal{N}$. Hence, L is also decidable in exponential time by some Ev$_2$-RNN[$\mathbb{R}$] as well as by some Ev-RNN[$\mathbb{R}$], namely by $\mathcal{N}$ itself. This shows that any language L can be decided in exponential time by some Ev$_2$-RNN[$\mathbb{R}$]s or some Ev-RNN[$\mathbb{R}$]s.

We now treat the case of the polynomial time of computation. Suppose that L is decidable in polynomial time by some Ev$_2$-RNN[$\mathbb{R}$] $\mathcal{N}$. Then, L is also decidable in polynomial time by some Ev-RNN[$\mathbb{R}$], namely by $\mathcal{N}$ itself. Furthermore, since Lemma 7 is originally stated for the case of Ev-RNN[$\mathbb{R}$]s, it follows that Lemma 11 – which appeals to Lemma 7 in its proof – can also be generalized in the context of

Ev-RNN[$\mathbb{R}$]s. Consequently, $L \in$ **P/poly**. This provides the two implications from Ev_2-RNN[$\mathbb{R}$]s and Ev-RNN[$\mathbb{R}$]s to **P/poly**.

Conversely, suppose that $L \in$ **P/poly**. By Theorem 12(a), L is decidable in polynomial time by some Ev_2-RNN[$\mathbb{Q}$] $\mathcal{N}$. Hence, L is also decidable in polynomial time by some Ev_2-RNN[$\mathbb{R}$] as well as by some Ev-RNN[$\mathbb{R}$], namely by $\mathcal{N}$. This provides the two implications from **P/poly** to Ev_2-RNN[$\mathbb{R}$]s and Ev-RNN[$\mathbb{R}$]s. □

Finally, Theorems 6, 12, 13, and 14 directly imply the super-Turing computational equivalence of the five models of Ev_2-RNN[$\mathbb{Q}$]s, Ev-RNN[$\mathbb{Q}$]s, Ev_2-RNN[$\mathbb{R}$]s, Ev-RNN[$\mathbb{R}$]s, and St-RNN[$\mathbb{R}$]s. This result shows that the super-Turing level of computation is achieved by the model of Ev_2-RNN[$\mathbb{Q}$]s, and that the incorporation of any more general patterns of evolvability or any possible real synaptic weights in this model does actually not further increase its computational capabilities.

Corollary 15. *The models of Ev_2-RNN[$\mathbb{Q}$]s, Ev-RNN[$\mathbb{Q}$]s, Ev_2-RNN[$\mathbb{R}$]s, Ev-RNN[$\mathbb{R}$]s, and St-RNN[$\mathbb{R}$]s are super-Turing equivalent to each other in polynomial time as well as in exponential time of computation.*

5.6 Discussion

In this chapter, we provided a characterization of the computational capabilities of several neural network models involved in a classical computational framework [171]. The Boolean, the static rational, the static real, and the evolving recurrent neural networks are computationally equivalent to finite state automata, Turing machines, and Turing machines with advices, respectively. These results are summarized in Table 2.

	Boolean	Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	B-RNN[Q]s	St-RNN[Q]s	Ev_2-RNN[Q]s	Ev-RNN[Q]s
	FSA	TMs	TM/poly(A)s	TM/poly(A)s
	REG	**P**	**P/poly**	**P/poly**
	[89, 119]	[155]	[24, 26]	[24, 26]
$\mathbb{R}$	B-RNN[R]s	St-RNN[R]s	Ev_2-RNN[R]s	Ev-RNN[R]s
	FSA	TM/poly(A)s	TM/poly(A)s	TM/poly(A)s
	REG	**P/poly**	**P/poly**	**P/poly**
	[89, 119]	[154]	[24, 26]	[24, 26]

Table 2 – Computational power of Boolean and sigmoidal recurrent neural networks according to the nature of their synaptic weights and patterns of evolvability. "FSA", "TMs", and "TM/poly(A)s" stand for "Finite State Automata", "Turing Machines", and "Turing Machines with Polynomially Bounded Advices". The complexity classes **REG**, **P**, and **P/poly** are those decided in polynomial time of computation by these respective models.

More precisely, the translation from the Boolean to the sigmoidal context dras-

tically increases the computational power of the networks. Besides, in the context of rational-weighted synaptic connections, the translation from the static to the bi-valued evolving framework does provide additional computational capabilities to the networks (cf. Theorem 5 and Corollary 15). By contrast, in the case of real-weighted synaptic connections, the translation from the static to the evolving framework does not bring any additional computational power to the networks (cf. Theorem 6 and Corollary 15). Finally, the four models of bi-valued evolving/general evolving rational-weighted/real-weighted neural networks are computationally equivalent.

Consequently, the evolving recurrent neural networks are computationally equivalent to the static real-weighted ones, irrespective of whether their evolving synaptic weights are modelled by rational or real numbers, and irrespective of whether their patterns of evolvability are restricted to bi-valued updates or expressed by any other more general form of updating [26, 149, 154] (cf. Corollary 15). These maximal computational capabilities of first-order neural networks correspond to those of the Turing machine with advice model. These considerations support the *Thesis of Analog Computation* formulated by Siegelmann and Sontag, which claims that no reasonable abstract analog device can be more powerful (in polynomial time) than first-order static analog recurrent neural networks [149, 154].

In view of these results, the consideration of either static real synaptic weights, on the one hand, or evolving synaptic weights, on the other hand, does equivalently lead to the emergence of super-Turing computational capabilities for the underlying neural networks (cf. Corollary 15). In the static real-weighted context, the extra-recursive power arises from the deeper and deeper precisions of the real weights which the networks can access throughout their computations (Theorem 6). In the evolving case, the extra capabilities emerge from the potential non-recursive patterns of evolvability to which the networks might be subjected (cf. proofs of Proposition 8 and Lemma 10). But even if the concept of the *power of the continuum* and the mechanism of *architectural evolvability* are mathematically equivalent in this sense, they are nevertheless conceptually distinct: while the power of the continuum remains a conceptualization of the mind, the evolving capabilities of the networks are, by contrast, observable in nature.

The possible achievement of super-Turing potentialities by evolving neural networks depends on the possibility for "nature" to realize non-recursive patterns of evolvability. Otherwise, the whole process could be simulated by some Turing machine. The question of the achievement of such non-recursive patterns of evolvability by biological neural networks lies beyond the scope of this work, and fits within the global issue of hypercomputation, with its proponents and opponents. For deeper philosophical considerations about hypercomputation, see for instance [46, 47, 126, 160, 161]. Overall, the assumptions that nature would not only follow preprogrammed patterns, that biological structures could involve non-recursive processes, like purely random phenomena for instance, would suffice to acknowledge the existence of hypercomputational capabilities, for these features cannot be simulated by the Turing machine model. But even with these premises accepted, the issue of the possibility to harness such hypercomputational capabilities remains open, and of paramount importance.

Overall, the presented results support the claim that the general mechanism of evolvability should be critically involved in the computational and dynamical capabilities of biological neural networks, and more generally, in the processing and coding of information in the brain. They provide a theoretical complement to the numerous experimental studies emphasizing the importance of the phenomenon of plasticity in the brain's information processing [1, 48, 69].

6 Interactive Computation

6.1 Introduction

Since as long ago as the late 40's, Wiener's considerations about cybernetics have suggested that system's information processing, in its general form, should involve as a key feature some "circular causal relationships" between the system and its environment [188]. An action of the system would generate some change in its environment, and that change would be reflected in the system in return (feedback), thus potentially affecting its future behavior. Such a machine that adapts its responses based on feedback would be a machine that learns. Along these lines, in the context of modern computation, the classical computational approach from Turing [171] has been argued to "no longer fully corresponds to the current notion of computing in modern systems" [97] – especially when it refers to bio-inspired complex information processing systems. In the brain (or in organic life in general), information is rather processed in an *interactive* way [61, 185]: previous experience must affect the perception of future inputs, and older memories may themselves change with response to new inputs.

Following these considerations, we initiated the study of the computational power of recurrent neural networks from the perspective of *interactive computation* [28]. In this context, we introduced a model of *interactive recurrent neural networks* involved in a sequential and continuous exchange of information with their environment. We showed that the rational-weighted interactive neural networks are computationally equivalent to interactive Turing machines, and realize the class of so-called recursive continuous ω-translations. The real-weighted and the evolving interactive recurrent neural networks are computationally equivalent to interactive Turing machines with advices and realize the class of continuous ω-translations [21, 25, 32, 35, 36]. Moreover, the interactive analog and evolving neural networks are universal, in the sense of capturing the computational capabilities of any possible interactive system [32, 36].

This chapter provides an overview of these results. Section 6.2 presents the general features of the interactive paradigm of computation. Section 6.3 introduces the concept of an interactive Turing machine and an interactive Turing machine with advice. Section 6.4 studies the computational power of interactive static and evolving recurrent neural networks. Finally, Section 6.5 offers some concluding remarks.

6.2 Interactive Computation

Historical Background

Interactive computation refers to the computational framework where systems may react or interact with each other as well as with their environment during the computation [61, 185]. This paradigm was theorized in contrast to classical computation [171] which rather proceeds in a function-based transformation of a given input into some corresponding output (closed-box and amnesic fashion), and has been argued to "no longer fully correspond to the current notions of computing in modern systems" [97]. Interactive computation also provides a particularly appropriate framework for the consideration of natural and bio-inspired complex information processing systems [35, 96, 97].

Wegner first proposed a foundational approach to interactive computation [185]. In his work, he claimed that "interaction is more powerful than algorithms", in the sense that computations performed in an interactive way are capable of handling a wider range of problems than those performed in a classical way, namely, by standard algorithms and Turing machines [185, 186].

In this context, Goldin et al. introduced the concept of a *persistent Turing machine* (PTM) as a possible extension of the classical Turing machine model to the framework of interactive computation [60, 65]. A *persistent Turing machine* consists of a multi-tape machine whose inputs and outputs are given as streams of tokens generated in a dynamical and sequential manner, and whose work tape and current computational state are kept preserved during the transition from one interactive step to the next, rather than being reinitialized. In this sense, a PTM computation is sequentially interactive and history dependent. Goldin et al. further provided a transfinite hierarchical classification of PTMs according to their expressive power, and established that PTMs are more expressive (in a precise sense) than amnesic PTMs (an extension of classical Turing machines in their context of interactive computation), and hence also than classical Turing machines [60, 65].

All these consideration led Goldin and Wegner to formulate the so-called *Sequential Interaction Thesis*, a generalization of the Church-Turing Thesis in the realm of interactive computation, claiming that "any sequential interactive computation can be performed by a persistent Turing machine" [62, 63, 64, 65]. They argue that this hypothesis, when combined with their result that PTMs are more expressive than classical TMs, provides a formal proof of Wegner's conjecture that "interaction is more powerful than algorithms" [62, 63, 64, 65], and hence refutes what they call the Strong Church-Turing Thesis – different from the original Church-Turing Thesis –, stating that any possible computation can be captured by some Turing machine, or in other words, that "models of computation more expressive than TMs are impossible" [63, 64].

On the basis of similar motivations, Van Leeuwen and Wiedermann proposed a slightly different interactive framework where a general system interacts with its environment by translating an incoming input stream of bits into a corresponding output stream of bit in a sequential manner [95, 98]. In their study, they restrict themselves to deterministic systems, and provide mathematical characteri-

zations of interactively computable relations, interactively recognizable sets of inputs streams, interactively generated sets of output streams, and interactively computable translations.

In this context, they introduced the concept of an *interactive Turing machine* (I-TM), a transposition of the classical Turing machine model to their interactive framework [96]. They further introduced the concept of *interactive Turing machine with advice* (I-TM/A) as a non-uniform computational model in the context of interactive computation [96, 100]. Interactive Turing machines with advice were proven to be strictly more powerful than interactive Turing machines without advice [100, Proposition 5] and [96, Lemma 1], and were shown to be computationally equivalent to several other non-uniform models of interactive computation, like sequences of interactive finite automata, site machines, web Turing machines [96, 100], and more recently, to interactive analog neural networks and interactive evolving neural networks [21, 25, 35].

These considerations led van Leeuwen and Wiedermann to formulate an *Interactive Extension of the Church-Turing Thesis* which states that "any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice" [100].

As opposed to Goldin and Wegner, van Leeuwen and Wiedermann consider that interactivity alone is not sufficient to break the Turing barrier. It rather consists of a different instead of a more powerful paradigm than the classical computational framework [96, 97, 99]. They write [97]:

> "From the viewpoint of computability theory, interactive computing e.g. with I-TMs does not lead to super-Turing computing power. Interactive computing merely extends our view of classically computable functions over finite domains to computable functions (translations) defined over infinite domains. Interactive computers simply compute something different from non-interactive ones because they follow a different scenario."

Here, we follow this point of view and adopt a similar approach to interactive computation as presented in [95, 98].

The Interactive Paradigm

The general interactive computational paradigm consists of a step by step exchange of information between a system and its environment [95, 98]. In order to capture the unpredictability of next inputs at any time step, the dynamically generated input streams need to be modeled by potentially infinite sequences of symbols (indeed, any interactive computation over a finite input stream can a posteriori be replayed in a non-interactive way producing the same output) [62, 97, 185].

Here, we consider a basic interactive computational scenario similar to that described in [95]. At every time step, the environment first sends a non-empty input bit to the system (full environment activity condition), the system next updates its current state accordingly, and then answers by either producing a corresponding output bit or remaining silent. In other words, the system is not obliged to pro-

vide corresponding output bits at every time step, but might instead stay silent for a while (to express the need of some internal computational phase before producing a new output bit), or even staying silent forever (to express the case that it has died). Consequently, after infinitely many time steps, the system will have received an infinite sequence of consecutive input bits and translated it into a corresponding finite or infinite sequence of not necessarily consecutive output bits. In the sequel, we assume that every interactive system is deterministic.

Formally, given some interactive deterministic system $\mathcal{S}$, for any infinite input stream $s \in \{0,1\}^\omega$, we define the corresponding output stream $o_s \in \{0,1\}^{\leq\omega}$ of $\mathcal{S}$ as the finite or infinite subsequence of (non-λ) output bits produced by $\mathcal{S}$ after having processed input s. The deterministic nature of $\mathcal{S}$ ensures that the output stream o_s is unique. In this way, any interactive system $\mathcal{S}$ realizes an ω-translation $\varphi_\mathcal{S} : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by $\varphi_\mathcal{S}(s) = o_s$, for each $s \in \{0,1\}^\omega$.

An ω-translation ψ is then called *interactively deterministically computable*, or simply *interactively computable* iff there exists an interactive deterministic system $\mathcal{S}$ such that $\varphi_\mathcal{S} = \psi$. Note that in this definition, we do absolutely not require for the system $\mathcal{S}$ to be driven by a Turing program nor to contain any computable component of whatever kind. We simply require that $\mathcal{S}$ is deterministic and performs ω-translations in conformity with our interactive paradigm, namely in a sequential interactive manner, as precisely described above.

Interactive Computable Functions

The specific nature of the interactive computational scenario imposes strong conditions on the ω-translations performed by interactive deterministic systems in general. In fact, it can be proven that any interactively computable ω-translation is necessarily continuous. This result will be used in the sequel.

Proposition 16. *Let ψ be some ω-translation. If ψ is interactively computable, then it is continuous.*

Proof. Let ψ be an interactively computable ω-translation. Then by definition, there exists a deterministic interactive system $\mathcal{S}$ such that $\varphi_\mathcal{S} = \psi$. Now, consider the function $f : \{0,1\}^* \to \{0,1\}^*$ which maps every finite word u to the unique corresponding finite word produced by $\mathcal{S}$ after exactly $|u|$ steps of computation over input stream u provided bit by bit. Note that the deterministic nature of $\mathcal{S}$ ensures that the finite word $f(u)$ is indeed unique, and thus that the function f is well-defined.

We show that f is monotone. Suppose that $u \sqsubseteq v$. It follow that $v = u \cdot (v - u)$. Hence, according to our interactive scenario, the output strings produced by $\mathcal{S}$ after $|v|$ time steps of computation over input stream v, namely $f(v)$, simply consists of the output strings produced after $|u|$ time steps of computation over input u, namely $f(u)$, followed by the output strings produced after $|v - u|$ time steps of computation over input $v - u$. Consequently, $f(u) \sqsubseteq f(v)$, and therefore f is monotone.

We now prove that the ω-translation $\varphi_\mathcal{S}$ performed by the interactive system $\mathcal{S}$ corresponds to the the "limit" (as described in Section 2) of the monotone function

f, i.e., that $\varphi_{\mathcal{S}} = f_\omega$. Towards this purpose, given some infinite input stream $s \in \{0,1\}^\omega$, we consider in turn the two possible cases where $\varphi_{\mathcal{S}}(s)$ is either an infinite or a finite word.

First, suppose that $\varphi_{\mathcal{S}}(s) \in \{0,1\}^\omega$. By definition, the word $\varphi_{\mathcal{S}}(s)$ corresponds to the output stream produced by $\mathcal{S}$ after having processed the whole infinite input s, and, for any $i \geq 0$, the word $f(s[0{:}i])$ corresponds to the output stream produced by $\mathcal{S}$ after $i+1$ time steps of computation over the input $s[0{:}i]$. According to our interactive scenario, $f(s[0{:}i])$ is a prefix of $\varphi_{\mathcal{S}}(s)$, for all $i \geq 0$ (indeed, once again, what has been produced by $\mathcal{S}$ on s after infinitely many time steps, namely $\varphi_{\mathcal{S}}(s)$, consists of what has been produced by $\mathcal{S}$ on $s[0{:}i]$ after $i+1$ time steps, namely $f(s[0{:}i])$, followed by what has been produced by $\mathcal{S}$ on $s - s[0{:}i]$ after infinitely many time steps). Moreover, since $\varphi_{\mathcal{S}}(s) \in \{0,1\}^\omega$, it means that the sequence of partial output strings produced by $\mathcal{S}$ on input s after i time steps of computation is not eventually constant, i.e., $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$. Hence, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathcal{S}}(s) \in \{0,1\}^\omega$ for all $i \geq 0$ and $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$ ensure that $\varphi_{\mathcal{S}}(s)$ is the unique infinite word containing each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix. This is to say by definition that $\varphi_{\mathcal{S}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_\omega(s)$.

Secondly, suppose that $\varphi_{\mathcal{S}}(s) \in \{0,1\}^*$. By the very same argument as in the previous case, $f(s[0{:}i])$ is a prefix of $\varphi_{\mathcal{S}}(s)$, for all $i \geq 0$. Moreover, since $\varphi_{\mathcal{S}}(s) \in \{0,1\}^*$, the sequence of partial output strings produced by $\mathcal{S}$ on input s after i time steps of computation must become stationary from some time step onwards, i.e. $\lim_{i\to\infty} |f(s[0{:}i])| < \infty$. Hence, the entire finite output stream $\varphi_{\mathcal{S}}(s)$ must necessarily have been produced after a finite amount of time, and thus $\varphi_{\mathcal{S}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$. Consequently, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathcal{S}}(s) \in \{0,1\}^*$ for all $i \geq 0$ and $\varphi_{\mathcal{S}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$ ensure that $\varphi_{\mathcal{S}}(s)$ is the smallest finite word that contains each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix. This is to say by definition that $\varphi_{\mathcal{S}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_\omega(s)$. Consequently, $\varphi_{\mathcal{S}}(s) = f_\omega(s)$ for any $s \in \{0,1\}^\omega$, meaning that $\varphi_{\mathcal{S}} = f_\omega$.

We proved that f is a monotone function satisfying $\varphi_{\mathcal{S}} = f_\omega$. This means by definition that $\varphi_{\mathcal{S}}$ is continuous. Since $\varphi_{\mathcal{S}} = \psi$, it follows that ψ is also continuous. □

6.3 Interactive Turing Machines

An *interactive Turing machine* consists of an interactive abstract device driven by a standard Turing machine program. It receives an infinite stream of bits as input and produces a corresponding stream of bits as output, step by step. The input and output bits are processed via corresponding input and output ports rather than tapes. Consequently, at every time step, the machine can no more operate on the output bits that have already been processed.[1] Furthermore, according to our interactive scenario, it is assumed that, at every time step, the environment sends a non-silent input bit to the machine, and the machine either answers by producing some corresponding output bit, or rather chooses to remain silent. The formal definition reads as follows.

[1] In fact, allowing the machine to erase its previous output bits would lead to the consideration of much more complicated ω-translations.

Definition 17. A deterministic *interactive Turing machine* (I-TM) $\mathcal{M}$ is defined as a tuple $\mathcal{M} = (Q, q_0, \Gamma, \delta)$, where:

- Q is a finite set of states;
- $q_0 \in Q$ is the initial state;
- $\Gamma = \{0, 1, \natural\}$ is the alphabet of the machine, where $\natural$ stands for the blank tape symbol;
- $\delta : Q \times \Gamma \times \{0, 1\} \to Q \times \Gamma \times \{\leftarrow, \rightarrow, -\} \times \{0, 1, \lambda\}$ is the transition function of the machine.

The relation $\delta(q, x, b) = (q', x', d, b')$ means that if the machine $\mathcal{M}$ is in state q, the cursor of the tape is scanning the letter $x \in \{0, 1, \natural\}$, and the bit $b \in \{0, 1\}$ is currently received at its input port, then $\mathcal{M}$ will go in next state q', it will make the cursor overwrite symbol x by $x' \in \{0, 1, \natural\}$ and then move to direction d, and it will finally output symbol $b' \in \{0, 1, \lambda\}$ at its output port, where λ represents the fact the machine is not outputting any bit at that time step. An interactive Turing machine is illustrated in Figure 20.

From this definition, any I-TM $\mathcal{M}$ induces an ω-translation $\varphi_{\mathcal{M}} : \{0, 1\}^\omega \to \{0, 1\}^{\leq\omega}$ mapping every infinite input stream s to the corresponding finite or infinite output stream o_s produced by $\mathcal{M}$. An ω-translation $\psi : \{0, 1\}^\omega \to \{0, 1\}^{\leq\omega}$ is said to be *realizable* by some interactive Turing machine iff there exists an I-TM $\mathcal{M}$ such that $\varphi_{\mathcal{M}} = \psi$.

Van Leeuwen and Wiedermann also introduced the concept of *interactive Turing machine with advice* as a relevant non-uniform computational model in the context of interactive computation [96, 100].

Formally, a deterministic *interactive Turing machine with advice* (I-TM/A) $\mathcal{M}$ consists of an interactive Turing machine provided with an advice mechanism, which comes in the form of an advice function $\alpha : \mathbb{N} \to \{0, 1\}^*$. In addition, the machine $\mathcal{M}$ uses two auxiliary special tapes, an advice input tape and an advice output tape, as well as a designated advice state. During its computation, $\mathcal{M}$ has the possibility to write the binary representation of an integer m on its advice input tape, one bit at a time. Yet at time step n, the number m is not allowed to exceed n. During the computation, if the machine happens to enter its designated advice state at some time step, then the string $\alpha(m)$ is written on the advice output tape in one time step, replacing the previous content of the tape. The machine has the possibility to repeat this process as many time as needed during its infinite computation. An interactive Turing machine with a advice is illustrated in Figure 21.

Once again, according to our interactive scenario, any I-TM/A $\mathcal{M}$ induces an ω-translation $\varphi_{\mathcal{M}} : \{0, 1\}^\omega \to \{0, 1\}^{\leq\omega}$ which maps every infinite input stream s to the corresponding finite or infinite output stream o_s produced by $\mathcal{M}$. Finally, an ω-translation $\psi : \{0, 1\}^\omega \to \{0, 1\}^{\leq\omega}$ is said to be *realizable* by some interactive Turing machine with advice iff there exists an I-TM/A $\mathcal{M}$ such that $\varphi_{\mathcal{M}} = \psi$.

For sake of completeness, we provide a proof that I-TM/A are strictly more powerful than I-TM. Accordingly, we say that I-TM/A are *super-Turing*. The result has already been mentioned in [100, Proposition 5] and [96, Lemma 1]

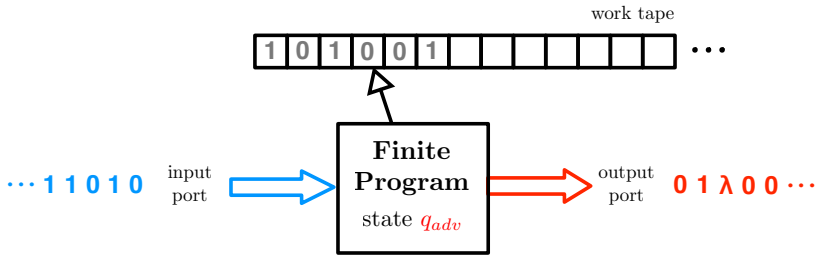


Figure 20 – An interactive Turing machine. The input and output streams of bits are processed via corresponding input and output ports rather than tapes.

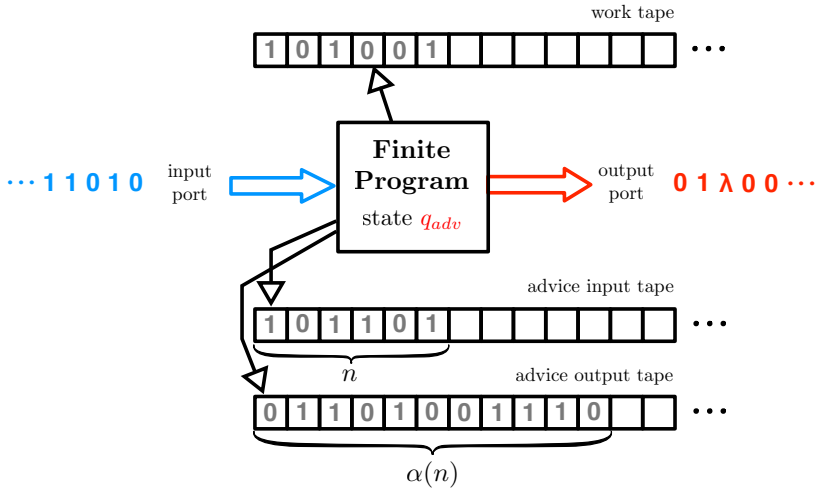


Figure 21 – An interactive Turing machine with advice. An advice input and output tapes are added to the interactive Turing machine.

Proposition 18. *I-TM/As are strictly more powerful than I-TMs.*

Proof. We prove that there exists an ω-translation ψ which is realizable by some I-TM/A, yet by no I-TM. Consider a non-Turing computable function $\alpha : \mathbb{N} \to \{0,1\}^*$. Note that such a function obviously exists since there are $2^{\aleph_0}$ (i.e. uncountably many) distinct functions of that form whereas there are only $\aleph_0$ (i.e. countably many) possible Turing machines. Consider the ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ which maps every infinite input stream s, necessarily writable of the form

$$s = 0^* b_0 0^+ b_1 0^+ b_2 0^+ b_3 \cdots$$

where b_i's denote the blocks of 1's occurring between the 0's, to the corresponding finite or infinite word given by

$$\psi(s) = \alpha(|b_0|)\alpha(|b_1|)\alpha(|b_2|)\alpha(|b_3|)\cdots$$

(if s has suffix 0^ω, then $\psi(s)$ is finite).

The ω-translation ψ is clearly realizable by some I-TM/A $\mathcal{M}$ with advice function α. Indeed, on every input stream $s \in \{0,1\}^\omega$, the machine $\mathcal{M}$ stores the successive blocks $b_0, b_1, b_2, \ldots$ of 1's occurring in s, and, for every such block b_i, first computes the length $|b_i|$, writes it in binary on its advice tape, then calls the advice value $\alpha(|b_i|)$ (or waits enough time steps in order to have the right to call it), and finally outputs the value $\alpha(|b_i|)$, before reiterating the procedure with respect to the next block b_{i+1}. In this way, $\mathcal{M}$ realizes ψ.

On the other hand, the ω-translation ψ is not realizable by any I-TM. Indeed, towards a contradiction, suppose it is realizable by some I-TM $\mathcal{M}$. Then, consider the classical Turing machine $\mathcal{M}'$ which, on every finite input r of the form $r = 1^k$, proceeds exactly like $\mathcal{M}$ would have on any infinite input beginning by r, thus outputs $\alpha(k)$, and diverges on every other finite input. The existence of this classical TM $\mathcal{M}'$ shows that the function α is Turing computable, a contradiction. □

Moreover, a precise characterization of the computational powers of I-TMs and I-TM/As can be given. In fact, the I-TMs and I-TM/As realize precisely the classes of recursive continuous and continuous ω-translations, respectively. The following results are proven in [25].

Proposition 19. *Let ψ be some ω-translation.*

1. *ψ is realizable by some I-TM iff ψ is recursive continuous.*
2. *ψ is realizable by some I-TM/A iff ψ is continuous.*

Proof. Point (1). Let ψ be some ω-translation realized by some I-TM $\mathcal{M}$. This means that $\psi = \varphi_{\mathcal{M}}$. Now, consider the function $f : \{0,1\}^* \to \{0,1\}^*$ which maps every finite word u to the unique corresponding finite word produced by $\mathcal{M}$ after exactly $|u|$ steps of computation over input stream u provided bit by bit. Since $\mathcal{M}$ is driven by a classical TM program, f is recursive. Moreover, by the exact same argument as in the proof of Proposition 16, f is monotone and $f_\omega = \varphi_{\mathcal{M}} = \psi$. Consequently, ψ is recursive continuous.

Conversely, let ψ be a recursive continuous ω-translation. Then there exists some recursive monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$. Now consider the infinite Procedure 3 (given in the proof of Theorem 21) where the instruction "decode $f(s[0{:}i])$ from r_f" is replaced by the recursive one: "compute $f(s[0{:}i])$". Since f is recursive, this slightly modified version of Procedure 3 can clearly be performed by an I-TM $\mathcal{M}$. The machine $\mathcal{M}$ outputs the current word $v - u$ bit by bit every time it reaches up the instruction "output $v - u$ bit by bit", and otherwise, keeps outputting λ symbols while simulating any other internal computational steps.

It remains to prove that the machine $\mathcal{M}$ realizes ψ, i.e. that $\varphi_{\mathcal{M}} = \psi$. Note that, for any input stream $s \in \{0,1\}^\omega$, the finite word that has been output at the end of each instruction "output $v - u$ bit by bit" corresponds precisely to the finite word $f(s[0{:}i])$ currently stored in the variable v. Hence, after infinitely many time steps, the finite or infinite word $\varphi_{\mathcal{M}}(s)$ output by $\mathcal{M}$ contains each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix. In other words, $f(s[0{:}i]) \subseteq \varphi_{\mathcal{M}}(s)$ for all $i \geq 0$.

We now consider in turn the two possible cases where $\varphi_{\mathcal{M}}(s)$ is either infinite or finite. First, if $\varphi_{\mathcal{M}}(s)$ is infinite, then it means that Procedure 3 has never stopped

outputting new bits from some time step onwards, i.e., $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$. Consequently, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathcal{M}}(s) \in \{0,1\}^{\omega}$ for all $i \geq 0$ and $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$ ensure that $\varphi_{\mathcal{M}}(s)$ is the unique infinite word containing each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix. This is to say by definition that $\varphi_{\mathcal{M}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_{\omega}(s)$. Second, if $\varphi_{\mathcal{N}}(s)$ is finite, it means that Procedure 3 has stopped outputting new bits from some time step onwards, and hence $\varphi_{\mathcal{M}}(s) = f(s[0{:}j])$ for some $j \geq 0$. In this case, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathcal{M}}(s) \in \{0,1\}^{*}$ for all $i \geq 0$ and $\varphi_{\mathcal{M}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$ ensure that $\varphi_{\mathcal{M}}(s)$ is the smallest finite word that contains each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix. Once again, this is to say by definition that $\varphi_{\mathcal{M}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_{\omega}(s)$.

Therefore, $\varphi_{\mathcal{M}} = f_{\omega}$, and since $f_{\omega} = \psi$, it follows that $\varphi_{\mathcal{M}} = \psi$, meaning that ψ is realized by $\mathcal{M}$.

Point (2). Let ψ be some ω-translation realized by some I-TM/A $\mathcal{M}$. By definition, ψ is interactively computable. By Proposition 16, ψ is continuous.

Conversely, let ψ be a continuous ω-translation. Then there exists some monotone function $f : \{0,1\}^{*} \to \{0,1\}^{*}$ such that $f_{\omega} = \psi$. First of all, we consider the function $\alpha : \mathbb{N} \to \{0,1\}^{*}$ which maps every integer n to the finite binary word $w_n \in \{0,1,2\}^{*}$ given by $w_n = 2 \cdot f(w_{n,1}) \cdot 2 \cdot f(w_{n,2}) \cdot 2 \cdots 2 \cdot f(w_{n,2^n}) \cdot 2$, where $w_{n,1}, \ldots, w_{n,2^n}$ denotes the lexicographical enumeration of all binary words of length n. Now consider the infinite Procedure 4 (given in the proof of Theorem 22) where the two instructions "access to the value q_{i+1}" and "decode $f(s[0{:}i])$ from q_{i+1}" are replaced by the two following ones: "query $\alpha(i+1) = w_{i+1}$" and "extract the subword $f(s[0{:}i])$ from w_{i+1}". This slightly modified version of Procedure 4 can clearly be performed by an I-TM/A $\mathcal{M}$ with advice function α (based on the extended alphabet $\Gamma = \{0,1,2,\sharp\}$). Every time $\mathcal{M}$ encounters the instruction "query $\alpha(i+1) = w_{i+1}$", it makes an extra-recursive call to its advice value $\alpha(i+1)$; otherwise, $\mathcal{M}$ simulates every other recursive step by means of its classical Turing program. Moreover, $\mathcal{M}$ outputs the current word $v - u$ bit by bit every time it reaches up the instruction "output $v - u$ bit by bit", and otherwise keeps outputting λ symbols while simulating any other internal computational steps. By the same argument as that presented in the end of Point (1), one has that $\varphi_{\mathcal{M}} = f_{\omega} = \psi$, meaning that ψ is realized by $\mathcal{M}$. □

6.4 Interactive Recurrent Neural Networks

6.4.1 The model

We consider a classical model of a first-order recurrent neural network, as described in Chapter 4, and transpose it in the context of interactive computation. In order to stay consistent with the interactive scenario presented in Section 6.2, our model of an *interactive recurrent neural network* (I-RNN) adheres to a rigid encoding of the way inputs and outputs are interactively processed between the environment and the network.

Formally, an *interactive recurrent neural network* (I-RNN) is a recurrent neural net that contains a finite number of internal neurons $(x_i)_{i=1}^{N}$, one Boolean input

neurons u, and two designated Boolean output neurons y_d and y_v. The role of the Boolean input cell u is to transmit to the network the infinite input stream of bits sent by the environment (the full environment activity condition forces that $u(t)$ never equals λ). The role of the Boolean data cell y_d is to carry the output stream of the network to the environment, while the role of the Boolean validation cell y_v is to describe when the data cell is active and when it is silent. Accordingly, the output stream transmitted by the network to the environment will be defined as the finite or infinite subsequence of successive data bits that occur simultaneously with positive validation bits.

The dynamics of the network is computed in the usual way: given the activation values of the input u and internal neurons and $(x_j)_{j=1}^N$ at time t, the activation values of each internal and output neuron x_i and y_i at time $t+1$ are updated by the following equations, respectively:

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + b_i(t)\cdot u(t) + c_i(t)\right) \text{ for } i = 1,\ldots,N \tag{6.1}$$

$$y_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + b_i(t)\cdot u(t) + c_i(t)\right) \text{ for } i = d,v \tag{6.2}$$

where $a_{ij}(t)$, $b_i(t)$, and $c_i(t)$ are the weights of the synaptic connections and the bias of the network at time t, and σ and θ are the sigmoid-linear and Heaviside step activation functions, respectively. An I-RNN is illustrated in Figure 22.

Let $\mathcal{N}$ be some I-RNN with its initial state set to zero, i.e. $\boldsymbol{x}(0) = \mathbf{0}$. Then any infinite input stream

$$s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$$

transmitted to input cell u induces via Equations (6.1) or (6.2) a corresponding pair of infinite streams transmitted by cells y_d and y_v

$$(y_d(0)y_d(1)y_d(2)\cdots, y_v(0)y_v(1)y_v(2)\cdots) \in \{0,1\}^\omega \times \{0,1\}^\omega.$$

The output stream of $\mathcal{N}$ associated to input s is then given by the finite or infinite subsequence o_s of successive data bits that occur simultaneously with positive validation bits, namely

$$o_s = \langle y_d(i) : i \in \mathbb{N} \text{ and } y_v(i) = 1\rangle \in \{0,1\}^{\leq\omega}.$$

Hence, any I-RNN $\mathcal{N}$ naturally induces an ω-translation $\varphi_\mathcal{N} : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by $\varphi_\mathcal{N}(s) = o_s$, for each $s \in \{0,1\}^\omega$. Finally, an ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ is said to be *realizable* by some I-RNN iff there exists some I-RNN $\mathcal{N}$ such that $\varphi_\mathcal{N} = \psi$.

In this work, four models of I-RNNs will be considered according to whether their underlying synaptic weights are either rational or real numbers of either static or evolving nature.

1. the *interactive static rational RNNs* (I-St-RNN[Q]s) refer to the class of all I-RNNs whose every weights are static and modelled by rational values.

2. the *interactive static real (or analog) RNNs* (I-St-RNN[$\mathbb{R}$]s) refer to the class of all I-RNNs whose every weights are static and modelled by real values.
3. the *interactive evolving rational RNNs* (I-Ev-RNN[$\mathbb{Q}$]s) refer to the class of all I-RNNs whose every evolving and static weights are rational.
4. the *interactive evolving real RNNs* (I-Ev-RNN[$\mathbb{R}$]s) refer to the class of all I-RNNs whose every evolving and static weights are real.

Since rational numbers are included in real numbers and since static weights are particular cases of evolving ones, the following inclusions hold by definition:

$$\begin{array}{ccc} \text{I-St-RNN}[\mathbb{Q}]\text{s} & \subsetneq & \text{I-Ev-RNN}[\mathbb{Q}]\text{s} \\ \rotatebox{90}{$\subsetneq$} & & \rotatebox{90}{$\subsetneq$} \\ \text{I-St-RNN}[\mathbb{R}]\text{s} & \subsetneq & \text{I-Ev-RNN}[\mathbb{R}]\text{s} \end{array}$$

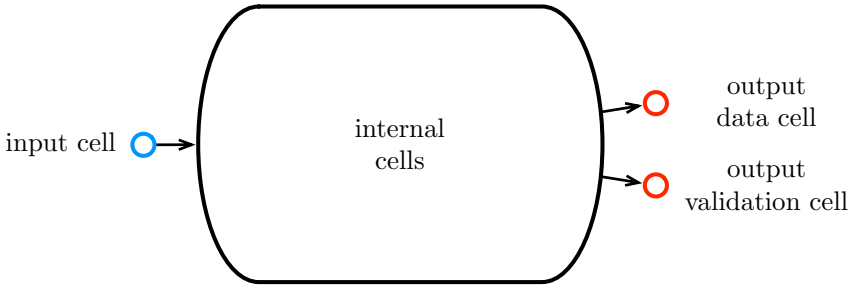


Figure 22 – Schematic representation of an interactive recurrent neural network.

6.4.2 Computational Power: The Static Case

In this section, we show that I-St-RNN[$\mathbb{Q}$]s and I-St-RNN[$\mathbb{R}$]s are computationally equivalent to I-TMs and I-TM/As and recognize the classes of recursive continuous and continuous ω-translations, respectively. Consequently, as for the classical case, the incorporation of the power of the continuum in the model does bring additional capabilities to the networks. In this sense, I-St-RNN[$\mathbb{R}$]s are super-Turing.

First, we consider the case if I-St-RNN[$\mathbb{Q}$]s.

Theorem 20. *I-St-RNN[$\mathbb{Q}$] are Turing-equivalent. More precisely, for any ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$, the following conditions are equivalent:*

1. *ψ is realizable by some I-St-RNN[$\mathbb{Q}$];*
2. *ψ is realizable by some I-TM;*
3. *ψ is recursive continuous.*

Proof. The classical equivalence between rational-weighted neural networks and Turing machines [155] can naturally be transposed in this interactive framework. This proves the equivalence "1 $\leftrightarrow$ 2". The equivalence "2 $\leftrightarrow$ 3" is provided by Proposition 19(1). □

Secondly, we consider the case if I-St-RNN[$\mathbb{R}$]s.

Theorem 21. *I-St-RNN[$\mathbb{R}$] are super-Turing. More precisely, for any ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$, the following conditions are equivalent:*

1. *ψ is realizable by some I-St-RNN[$\mathbb{R}$];*
2. *ψ is realizable by some I-TM/A;*
3. *ψ is continuous.*

Proof. The equivalence "2 $\leftrightarrow$ 3" is provided by Proposition 19(2). The implication "1 $\to$ 3" is given by Proposition 16. We prove the remaining converse implication "3 $\to$ 1".

Let $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ be some continuous ω-translation. Then there exists some monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such such that $f_\omega = \psi$. We begin by recursively encoding all successive values of f into some real number r_f. Let $(w_n)_{n\in\mathbb{N}}$ be the lexicographical enumeration of all binary words, i.e., $w_0 = \lambda$, $w_1 = 0$, $w_2 = 1$, $w_3 = 00$, $w_4 = 01$, etc., and consider the infinite word $w_f \in \{0,1,2\}^\omega$ given by $w_f = 2 \cdot f(w_0) \cdot 2 \cdot f(w_1) \cdot 2 \cdot f(w_2) \cdot 2 \cdots$. Then, we consider the real encoding r_f of the word w_f given by

$$r_f = \sum_{i=1}^{\infty} \frac{2 \cdot w_f(i) + 1}{6^i}.$$

One has $r_f \in]0,1[$, and for any $n \in \mathbb{N}$, it can be shown that the finite word $f(w_n)$ can be decoded from the value r_f by some Turing machine, or equivalently, by some rational recurrent neural network [154, 155].

Now, we consider the infinite Procedure 3 described below. This procedure receives as input an infinite stream $s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$ provided bit by bit, and eventually produces as output a corresponding finite or infinite stream of bits. The procedure consists of two infinite subroutines running in parallel. The first subroutine stores each input bit $s(t)$ occurring at every time step t. The second subroutine performs an infinite loop. More precisely, at stage $i+1$, the procedure considers the value $f(s[0{:}i{+}1])$. By monotonicity of f, the word $f(s[0{:}i{+}1])$ extends $f(s[0{:}i])$. If this extension is strict, the procedure output the difference word $f(s[0{:}i{+}1]) - f(s[0{:}i])$ bit by bit. Otherwise, the procedure simply outputs the empty word λ. Note that every instruction of Procedure 3 is recursive provided that the real number r_f is given in advance.

We show that there indeed exists some I-St-RNN[$\mathbb{R}$] $\mathcal{N}$ which can simulate Procedure 3. The network $\mathcal{N}$ consists a rational-weighed neural net connected to only one other neuron x_f with background activity of real intensity r_f. The neuron x_f provides the possibility to access to the real number r_f at any time, via its background activity, and the remaining rational-weighted net is designed in order to perform all the recursive steps of Procedure 3. The equivalence between rational-weighted RNNs and TMs ensures that such a static rational-weighted net can always be constructed [154]. In addition, the network $\mathcal{N}$ is designed in such a way that it outputs via its data and validation cells y_d and y_v the finite word $v - u$ every

Procedure 3 : uses the designated real number r_f

input: infinite input stream $s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$ provided bit by bit
initialization: $i \leftarrow 0$, $x \leftarrow \lambda$, $u \leftarrow \lambda$, $v \leftarrow \lambda$

SUBROUTINE 1:
for all $t \geq 0$ **do**
  $x \leftarrow x \cdot s(t)$ `// concatenation of the current bit s(t) to x`
end for

SUBROUTINE 2:
loop
  decode $s[0{:}i]$ from x
  decode $f(s[0{:}i])$ from r_f `// recursive instruction if r_f is given in advance`
  $v \leftarrow f(s[0{:}i])$
  if $u \subsetneq v$ **then**
    output $v - u$ bit by bit
  else
    output λ
  end if
  $i \leftarrow i + 1$
  $u \leftarrow v$
end loop

time it simulates the instruction "output $v - u$ bit by bit" of Procedure 3. The network keeps outputting λ symbols while simulating any other internal instruction of Procedure 3.

Finally, a direct transposition of the argument presented in the proof of Proposition 19(1) shows that $\mathcal{N}$ realizes ψ, i.e. that $\varphi_{\mathcal{N}} = \psi$. This concludes the proof. □

6.4.3 Computational Power: The Evolving Case

In this section, we prove that interactive evolving RNNs are super-Turing, irrespective of whether their synaptic weights are modeled by rational or real numbers. More precisely, both models of interactive rational and interactive real RNNs are computationally equivalent to interactive Turing machines with advice, and thus realize the class of continuous ω-translations. Consequently, in both classical and interactive frameworks, the translation from static rational to the evolving rational context does bring additional computational power to the networks; by contrast, the translation from the evolving rational to the evolving real does not increase their capabilities.

Theorem 22. *I-Ev-RNN[$\mathbb{Q}$]s and I-Ev-RNN[$\mathbb{R}$]s are super-Turing. More precisely, for any ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$, the following conditions are equivalent:*

1. *ψ is realizable by some I-Ev-RNN[$\mathbb{Q}$];*
2. *ψ is realizable by some I-Ev-RNN[$\mathbb{R}$];*
3. *ψ is realizable by some I-TM/A;*

4. *ψ is continuous.*

Proof. The implication "1 $\to$ 2" holds by definition. The three implications "1 $\to$ 4", "2 $\to$ 4", and "3 $\to$ 4" are given by Proposition 16. The equivalence "4 $\leftrightarrow$ 3" is provided by Proposition 19(2).

We now prove the last remaining implication "4 $\to$ 1". The argument closely resembles that of Theorem 21. For sake of completeness, we chose provide it. Let ψ be a continuous function. Then there exists some monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such such that $f_\omega = \psi$. In this case, all possible values of f into successive distinct rational numbers. Towards this purpose, for any $n > 0$, we let $w_{n,1}, \ldots, w_{n,2^n}$ be the lexicographical enumeration of all binary words of length n, and we let $w_n \in \{0,1,2\}^*$ be the finite word given by $w_n = 2 \cdot f(w_{n,1}) \cdot 2 \cdot f(w_{n,2}) \cdot 2 \cdots 2 \cdot f(w_{n,2^n}) \cdot 2$. Then, we consider the following rational encoding of the word w_n

$$q_n = \sum_{i=1}^{|w_n|} \frac{2 \cdot w_n(i) + 1}{6^i}.$$

Note that $q_n \in]0,1[$ for all $n > 0$. Also, the encoding procedure ensures that $q_n \neq q_{n+1}$, since $w_n \neq w_{n+1}$, for all $n > 0$. Moreover, it can be shown that the finite word w_n can be decoded from the value q_n by some Turing machine, or equivalently, by some rational recurrent neural network [154, 155]. In this way, for any $n > 0$, the number q_n provides a rational encoding of the images by f of all words of length n.

Now, we consider the infinite Procedure 4 described below. Note that the only non-recursive instruction of Procedure 4 is "access to the value q_{i+1}".

Procedure 4 : involves a non-recursive instruction

input: infinite input stream $s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$ provided bit by bit
initialization: $i \leftarrow 0$, $x \leftarrow \lambda$, $u \leftarrow \lambda$, $v \leftarrow \lambda$

SUBROUTINE 1:
for all $t \geq 0$ **do**
  $x \leftarrow x \cdot s(t)$ // concatenation of the current bit $s(t)$ to x
end for

SUBROUTINE 2:
loop
  decode $s[0{:}i]$ from x
  access to the value q_{i+1} // non-recursive instruction
  decode $f(s[0{:}i])$ from q_{i+1}
  $v \leftarrow f(s[0{:}i])$
  if $u \subsetneq v$ **then**
    output $v - u$ bit by bit
  else
    output λ
  end if
  $i \leftarrow i + 1$
  $u \leftarrow v$
end loop

We show that there exists some I-Ev-RNN[Q] $\mathcal{N}$ which performs Procedure 4. The network $\mathcal{N}$ consists of one evolving and one static rational sub-network connected together. The evolving sub-network will be in charge of the execution of the only non-recursive instruction "access to the value q_{i+1}", and the static sub-network will be in charge of the execution of all other recursive instructions of Procedure 4.

More precisely, the evolving rational-weighted part of $\mathcal{N}$ is made up of a single designated processor x_e. The neuron x_e receives as sole incoming synaptic connection a background activity of evolving intensity $c_e(t)$. The synaptic weight $c_e(t)$ successively takes the rational bounded values $q_1, q_2, q_3, \ldots$, by switching from value q_k to q_{k+1} after every N_k time steps, for some large enough $N_k > 0$ to be described. In this way, every time some new value q_{i+1} appears as a background activity of neuron x_e, the network stores it in a designated neuron in order to be able to perform the instruction "access to the value q_{i+1}" when required.

The static rational-weighted part of $\mathcal{N}$ is designed in order to perform the successive recursive steps of Procedure 4, every time some new value q_{i+1} has appeared by means of the activation value of neuron x_e. The equivalence result between rational-weighted RNNs and TMs ensures that such a static rational-weighted sub-network of $\mathcal{N}$ performing these recursive step can be constructed [154]. Moreover, for each $k > 0$, the time interval N_k between the apparition of the synaptic weights q_k and q_{k+1} is chosen large enough in order to be able to perform all the aforementioned recursive steps.

In addition, the network $\mathcal{N}$ is designed in such a way that it outputs via its data and validation cells y_d and y_v the finite word $v - u$ every time it simulates the instruction "output $v - u$ bit by bit" of Procedure 4. The network keeps outputting λ symbols while simulating any other internal instruction of Procedure 4.

Finally, a direct transposition of the argument provided in the proof of Proposition 19(1) shows that $\mathcal{N}$ realizes ψ, i.e. that $\varphi_{\mathcal{N}} = \psi$. This concludes the proof. □

6.4.4 Universality

Theorems 21 and 22 together with Proposition 16 show that the four models of I-St-RNN[$\mathbb{R}$]s, I-Ev-RNN[Q]s, I-Ev-RNN[$\mathbb{R}$]s, and I-TM/As are capable to capture all possible computations performable by some deterministic interactive system. More precisely, for any possible interactive deterministic systems $\mathcal{S}$, there exists an I-St-RNN[$\mathbb{R}$] $\mathcal{N}_1$, an I-Ev-RNN[Q] $\mathcal{N}_2$, an I-Ev-RNN[$\mathbb{R}$] $\mathcal{N}_3$, and an I-TM/A $\mathcal{M}$ such that $\varphi_{\mathcal{N}_1} = \varphi_{\mathcal{N}_2} = \varphi_{\mathcal{N}_3} = \varphi_{\mathcal{M}} = \varphi_{\mathcal{S}}$. In this sense, those four models of interactive computation are called *universal*.

Theorem 23. *The four models of computations that are I-St-RNN[$\mathbb{R}$]s, I-Ev-RNN[Q]s, I-Ev-RNN[$\mathbb{R}$]s, and I-TM/As, are super-Turing universal.*

Proof. Let $\mathcal{S}$ be some deterministic interactive system. By Proposition 16, $\varphi_{\mathcal{S}}$ is continuous. By Theorems 21 and 22, $\varphi_{\mathcal{S}}$ is realizable by some I-St-RNN[$\mathbb{R}$], by some I-Ev-RNN[Q], by some I-Ev-RNN[$\mathbb{R}$], and by some I-TM/A. □

These results are summarized in Table 3. They can be understood as follows: similarly to the classical framework, where every possible partial function from

integers to integers can be computed by some Turing machine with oracle [171], in the interactive framework, every possible ω-translation performed in an interactive way can be computed by some interactive Turing machine with advice, or equivalently, by some interactive analog or evolving recurrent neural network.

6.5 Discussion

We showed that interactive rational- and real-weighted RNNs are Turing-equivalent and super-Turing, respectively (Theorems 20, 21). Furthermore, interactive evolving RNNs are also super-Turing, irrespective of whether their synaptic weights are modeled by rational or real numbers (Theorem 22). These results are summarized in Table 3.

	Static	Evolving
$\mathbb{Q}$	I-St-RNN[Q]s	I-Ev-RNN[Q]s
	I-TMs	I-TM/As
	recursive continuous	continuous
$\mathbb{R}$	I-St-RNN[$\mathbb{R}$]s	I-Ev-RNN[$\mathbb{R}$]s
	I-TM/As	I-TM/As
	continuous	continuous

Table 3 – Computational power of the four models of I-RNNs.

These results provide a direct generalization of those presented in previous Chapter 5. They show that in both classical and interactive computational frameworks, the translations from the static rational to the static real context, as well as from the static rational to the evolving rational context, do bring additional power to the underlying neural networks. By contrast, the two other translations from the evolving rational to the evolving real context, as well as from the static real to the evolving real context, do not increase further the capabilities of the networks.

Furthermore, according to Corollary 15 and Theorems 20, 21 and 22, the computational capabilities of all neural models studied so far are shown to be upper bounded by those of the Turing machine with advice model. In the classical computational context, these considerations support the *Thesis of Analog Computation*, which states that every natural computational phenomenon can be captured by the Turing machine with polynomially bounded advice model [149, 154]. In the interactive framework, they support the *Church-Turing Thesis of Interactive Computation*, which claims that "any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice" [100].

These considerations, together with those of Sections 5.4 and 5.5, support further the claim that the Turing machine with advice model could encompass the potentialities of brain computation, or even of natural computation in general [19, 97, 149], for it is able to capture crucial features, like analogue considerations [155],

evolvability [21, 24, 35], chaotic behaviors [168], that are impossible to be achieved via the simple Turing machine model.

The results also show once again that the incorporation of either *evolving capabilities* or of the *power of the continuum* in a basic neural model provides alternative and equivalent ways towards the achievement of maximal super-Turing computational capabilities. They support further the claim that the biological mechanism of plasticity should be crucially involved in the computational and dynamical capabilities of biological neural networks.

7 ATTRACTOR-BASED COMPUTATION

7.1 INTRODUCTION

In the central nervous system, one neuron may receive and send projections from and to thousands of other neurons. The huge number of connections established by a single neuron and the slow integration time of neurons, operating in the milliseconds range (billion times slower than presently available supercomputers), suggest that information in the nervous system might be transmitted by simultaneous discharges of large sets of neurons. In particular, the activation of functional cell assemblies in distributed networks might be induced by transmissions of complex patterns of activity [2, 3]. In this context, the temporal coding approach to neural information processing states that precise spike timing is a significant element in neural coding [20, 165]. Apart from the firing rate of the neural spikes, the *spatiotemporal pattern of discharges* – i.e., ordered and precise interspike interval relationships – should be significantly involved in the processing and coding of information in the brain, see [2, 3, 4, 5, 80, 116, 137, 174, 175, 178, 180] as well as the survey by Villa [176] and the references therein. Besides, *attractor dynamics* or *quasi-attractor dynamics* have been associated to perceptions, thoughts and memories, and the *chaotic intinerancy* between those with sequences in thinking, speaking and writing [79, 84, 166, 167, 168]. Specific chaotic attractor dynamics associated to spike series have been experimentally observed [39, 40, 177]. Furthermore, the correlation between attractor dynamics and repeating spatiotemporal firing patterns has been observed in simulations of nonlinear dynamical systems [14, 15] as well as in simulations of large scale neuronal networks [77, 78]. Consequently, the spatiotemporal patterns could be the witnesses of an underlying attractor dynamics.

These experimental considerations suggest that some aspect of the computational capabilities of neural networks are likely to be correlated to their attractor dynamics and spatiotemporal patterns of discharges. Accordingly, we initiated the study of the expressive power of recurrent neural networks from the perspective of their attractor dynamics. First of all, we focused on the case of Boolean recurrent neural networks, i.e., composed with Boolean input, output, and internal units. The Boolean input and output cells carry out the exchange of discrete information between the networks and their environment. When subjected to some infinite binary input stream, the Boolean output cells would necessarily enter into some attractor dynamics, and accordingly, elicit some corresponding spatiotemporal pattern of discharge. We generally assumed that the attractors can be of two possible types: meaningful or spurious. The neural ω-language of a network corresponds to the

set of all those input streams that induce a meaningful attractor dynamics. The expressive power of the networks is then measured via the topological complexity of their underlying neural ω-languages [27, 28, 29].

As a first step, we considered some output-related condition for the type specification of the networks' attractors: an attractor is defined as meaningful if it involves spiking cells from the output layer; it is spurious otherwise. In this case, the Boolean networks disclose the same expressive power as Büchi automata. As a second step, we assumed that the attractors' type specification is determined by external neurobiological criteria, instead of being associated to the output layers of the networks. Under this relaxed condition, the networks increase their computational capabilities from Büchi to Muller automata. Consequently, these Boolean networks recognize the class of all ω-regular neural languages. The most refined topological classification of ω-regular languages [182] can therefore be transposed from the automaton to the neural network context, and in turn, yield to some transfinite hierarchical classifications of Boolean neural networks based on their attractor dynamics. This hierarchical classification naturally induces some novel attractor-based measure of complexity for Boolean neural networks. This complexity measure is linked to the intricacy of the attractors' structure of the networks, or more precisely, to the maximal number of times that a network might alternate between meaningful and spurious attractors along some computation. This feature notably refers to the ability of the networks to perform more or less complicated classification tasks of their input streams via the manifestation of meaningful or spurious attractor dynamics. As an illustration, we computed the attractor-based complexity of a Boolean model of the basal-ganglia thalamocortical network.

As a next step, we extended the whole approach the context of sigmoidal (rather than Boolean) recurrent neural networks. In this case, we considered classical first-order recurrent neural network composed with Boolean input and output cells as well as sigmoidal internal units, along the lines of [24, 26, 154, 155]. The sigmoidal internal neurons introduce the source of nonlinearity which is so important to neural computation. They provide the possibility to surpass the capabilities of finite state automata, or even of Turing machines. Here again, the expressive power of the networks is measured via the topological complexity of the underlying neural ω-languages. This expressive power also refers to the ability of the networks to perform more or less complicated classification tasks of their input streams via the manifestation of meaningful or spurious attractor dynamics [22, 23, 30, 31, 34].

We first focused on the deterministic context. We showed that the static rational-weighted neural networks are computationally equivalent to the deterministic Muller Turing machines, and hence, recognize a class of neural ω-languages strictly inside the $BC(\mathbf{\Pi}_2^0)$-sets (the finite Boolean combinations of $\mathbf{\Pi}_2^0$-sets). We further proved that the static real-weighted neural networks and the general/bi-valued rational/real evolving neural networks are computationally equivalent. These models of static real and evolving neural networks are strictly more powerful than the deterministic Muller Turing machines and recognize the class of all $BC(\mathbf{\Pi}_2^0)$ neural ω-languages. Therefore, in this context again, analog and evolving neural networks are super-Turing.

We then investigated the nondeterministic context, and two specific forms of

nondeterminism are considered [22, 23]. In the first case, nondeterminism is expressed as an external binary guess stream processed by means of an additional Boolean guess cell, along the very lines of [154, 155]. In the second case, nondeterminism is expressed as a set of possible evolving patterns that the synaptic connections of the network might follow over the successive time steps [22]. At the beginning of a computation, the network selects one such possible evolving pattern – in a nondeterministic manner – and then sticks to it throughout its whole computational process. Six neural models of type 1 and four of type 2 are considered, according to the nature of their synaptic weights: static or evolving and rational or real. Overall, we proved that the static rational-weighted neural networks of type 1 are computationally equivalent to the nondeterministic Muller Turing machines. They recognize the class of all effectively analytic (i.e., Σ_1^1 lightface) sets. The nine other models of analog and evolving neural networks of types 1 and 2 are all computationally equivalent to each other, and strictly more powerful than the nondeterministic Muller Turing machines. They recognize the class of all analytic (i.e., $\mathbf{\Sigma_1^1}$ boldface) sets.

In both deterministic and nondeterministic cases, the complexity of the networks also refers to their ability to perform more or less complicated classification tasks of their input streams via the manifestation of meaningful or spurious attractor dynamics. The higher the topological complexity of the neural ω-language, the more complex the classification task.

In this chapter, we review this whole study of the expressive capabilities of recurrent neural networks in relation to their attractor dynamics. In Section 7.2, we recall the concepts of automata and Turing machines working on infinite input streams. In Section 7.3, we present the capabilities of Boolean recurrent neural networks. In Sections 7.4 and 7.5, we review the expressive powers of deterministic and nondeterministic sigmoidal recurrent neural networks. Finally, Section 7.6 provides some concluding remarks.

7.2 ω-AUTOMATA AND ω-TURING MACHINES

The study of the behavior of reactive systems has led to the emergence of a theory of automata working on infinite objects [134, 164]. In this section, we recall the concepts of automata and Turing machines provided with Büchi and Muller acceptance conditions.

ω-AUTOMATA

A *finite deterministic Büchi automaton* [134] is a 5-tuple $\mathcal{A} = (Q, A, i, \delta, \mathcal{F})$ where Q is a finite set of states, A is a finite alphabet, $i \in Q$ is the initial state, $\delta : Q \times A \to Q$ is a partial transition function, $\mathcal{F} \subseteq Q$ is a set of final states.

An infinite initial path ρ of $\mathcal{A}$ is called *successful* if it visits at least one of the final states infinitely often, i.e. if $\inf(\rho) \cap \mathcal{F} \neq \emptyset$. An infinite word is *recognized* by $\mathcal{A}$ if it is the label of a successful infinite path in $\mathcal{A}$. The *language recognized by* $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of all infinite words recognized by $\mathcal{A}$.

A *cycle* in $\mathcal{A}$ consists of a finite set of states c such that there exists a finite path

in $\mathcal{A}$ with same initial and final state and visiting precisely all states of c. A cycle c_j is *accessible* from cycle c_i if there exists a path from some state of c_i to some state of c_j. Furthermore, a cycle is *successful* if it contains a state belonging to $\mathcal{F}$, and *non-successful* otherwise.

An *alternating chain of length* $n \in \mathbb{N}$ (respectively *co-alternating chain of length* $n \in \mathbb{N}$) in $\mathcal{A}$ is a finite sequence of $n+1$ distinct cycles $(c_0, \ldots, c_n)$ such that:

- c_0 is successful (resp. c_0 is non-successful);
- c_i is successful iff c_{i+1} is non-successful;
- c_{i+1} is accessible from c_i, and c_i is not accessible from c_{i+1}, for all $i < n$.

An *alternating chain of length* ω in $\mathcal{A}$ is a sequence of two cycles (c_0, c_1) such that[1]:

- c_0 is successful;
- c_1 is non-successful;
- c_0 is accessible from c_1, and c_1 is also accessible from c_0;

In this case, cycles c_0 and c_1 are said to communicate. For any $\alpha \leq \omega$, an alternating chain of length α is said to be *maximal* in $\mathcal{A}$ if there is no alternating chain and no co-alternating chain in $\mathcal{A}$ of strictly larger length. A co-alternating chain of length α is said to be maximal in $\mathcal{A}$ if exactly the same condition holds.

These concepts are illustrated in Figure 23.

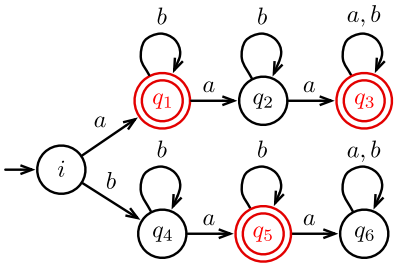


Figure 23 – A deterministic Büchi automaton $\mathcal{A}$. The double-circled red nodes correspond to the final states of $\mathcal{A}$. The language $L(\mathcal{A})$ recognized by $\mathcal{A}$ can be described by the following ω-regular expression $L(\mathcal{A}) = a(b^\omega + b^*ab^*a(a+b)^\omega) + bb^*ab^\omega$. Besides, note that every singleton $\{q_i\}$ is a cycle in $\mathcal{A}$. Cycles $\{q_1\}$, $\{q_3\}$ and $\{q_5\}$ are successful, since they all contain final states. Cycles $\{q_2\}$, $\{q_4\}$ and $\{q_6\}$ are by contrast non-successful. The automaton $\mathcal{A}$ contains maximal alternating and co-alternating chains of length 2 given by $(\{q_1\}, \{q_2\}, \{q_3\})$ and $(\{q_4\}, \{q_5\}, \{q_6\})$, respectively.

A *finite deterministic Muller automaton* is a 5-tuple $\mathcal{A} = (Q, A, i, \delta, \mathcal{T})$, where Q is a finite set of states, A is a finite alphabet, $i \in Q$ is the initial state, $\delta : Q \times A \to Q$ is a partial transition function, $\mathcal{T} \subseteq \mathcal{P}(Q)$ is a collection of set of states called the table of the automaton.

[1] We recall that ω denotes the least infinite ordinal.

In this case, infinite initial path ρ of $\mathcal{A}$ is now called *successful* if $\inf(\rho) \in \mathcal{T}$. Given a finite deterministic Muller automaton $\mathcal{A} = (Q, A, i, \delta, \mathcal{T})$, a cycle in $\mathcal{A}$ is *successful* if it belongs to $\mathcal{T}$, and *non-succesful* otherwise. An infinite word is *recognized* by $\mathcal{A}$ if it is the label of a successful infinite path in $\mathcal{A}$, and the *ω-language recognized by $\mathcal{A}$*, denoted by $L(\mathcal{A})$, is the set of all infinite words recognized by $\mathcal{A}$.

It can be shown that deterministic Muller automata are strictly more powerful than deterministic Büchi automata, but have an equivalent expressive power as nondeterministic Büchi automata, Rabin automata, Street automata, parity automata, and nondeterministic Muller automata. They recognize precisely the class of *ω-rational languages* [134].

For each ordinal α such that $0 < \alpha < \omega^\omega$, we introduce the concept of an *alternating tree* of length α in a deterministic Muller automaton $\mathcal{A}$, which consists of a tree-like disposition of the successful and non-successful cycles of $\mathcal{A}$ induced by the ordinal α, as illustrated in Figure 24. In order to describe this tree-like disposition, we first recall that any ordinal $0 < \alpha < \omega^\omega$ can uniquely be written of the form $\alpha = \omega^{n_p} \cdot m_p + \omega^{n_{p-1}} \cdot m_{p-1} + \ldots + \omega^{n_0} \cdot m_0$, for some $p \geq 0$, $n_p > n_{p-1} > \ldots > n_0 \geq 0$, and $m_i > 0$. Then, given some deterministic Muller automata $\mathcal{A}$ and some strictly positive ordinal $\alpha = \omega^{n_p} \cdot m_p + \omega^{n_{p-1}} \cdot m_{p-1} + \ldots + \omega^{n_0} \cdot m_0 < \omega^\omega$, an *alternating tree* (respectively *co-alternating tree*) of length α is a sequence of cycles of $\mathcal{A}$ $(C^{i,j}_{k,l})_{i \leq p, j < 2^i, k < m_i, l \leq n_i}$ such that:

- $C^{0,0}_{0,0}$ is successful (respectively non-successful);
- $C^{i,j}_{k,l} \subsetneq C^{i,j}_{k,l+1}$, and $C^{i,j}_{k,l+1}$ is successful iff $C^{i,j}_{k,l}$ is non-successful;
- $C^{i,j}_{k+1,0}$ is accessible from $C^{i,j}_{k,0}$, and $C^{i,j}_{k+1,0}$ is successful iff $C^{i,j}_{k,0}$ is non-successful;
- $C^{i+1,2j}_{0,0}$ and $C^{i+1,2j+1}_{0,0}$ are both accessible from $C^{i,j}_{m_i-1,0}$, and each $C^{i+1,2j}_{0,0}$ is successful whereas each $C^{i+1,2j+1}_{0,0}$ is non-successful.

An alternating tree of length α is said to be maximal in $\mathcal{A}$ if there is no alternating or co-altenrating tree in $\mathcal{A}$ of length $\beta > \alpha$. A co-alternating tree of length α is said to be maximal in $\mathcal{A}$ if exactly the same condition holds. An alternating tree of length α is illustrated in Figure 24.

These concepts are illustrated in Figure 25.

ω-Turing Machines

A *Büchi Turing machine* and a *Muller Turing machine* can be defined as a pair $(\mathcal{M}, \mathcal{F})$ and $(\mathcal{M}, \mathcal{T})$, respectively, where:

- $\mathcal{M}$ is a classical multitape Turing machine whose input tape is associated with a one way read-only head;
- $\mathcal{F}$ is a collection of states of $\mathcal{M}$;
- $\mathcal{T}$ is a collection of sets of states of $\mathcal{M}$, i.e., $\mathcal{T} = \{T_1, \ldots, T_k\}$ and each T_i is a set of states of $\mathcal{M}$.

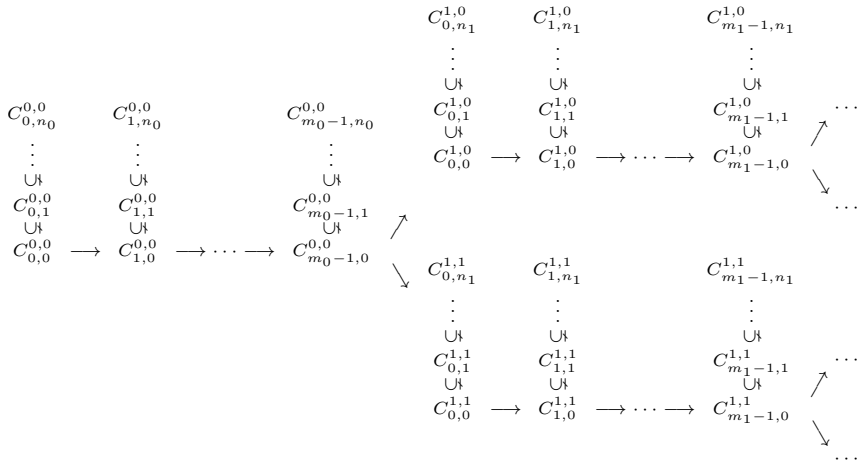
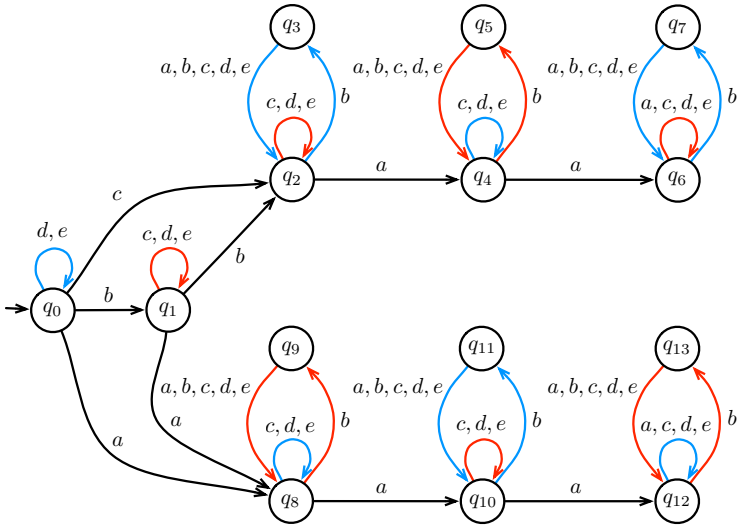


Figure 24 – Inclusion and accessibility relations between cycles of an alternating tree of length α, with $0 < \alpha < \omega^\omega$.



$$\mathcal{T} = \{\{q_0\}, \{q_2, q_3\}, \{q_4\}, \{q_6, q_7\}, \{q_8\}, \{q_{10}, q_{11}\}, \{q_{12}\}\}$$

sdkfksfd

Figure 25 – A Muller automaton $\mathcal{A}$. The underlying alphabet of $\mathcal{A}$ is $\{a, b, c, d, e\}$. The successful and non-successful cycles are denoted in blue and red, respectively. Let us consider the following cycles: $C^{0,0}_{0,0} = \{q_0\}$, $C^{0,0}_{1,0} = \{q_1\}$, $C^{1,0}_{0,0} = \{q_2\}$, $C^{1,0}_{0,1} = \{q_2, q_3\}$, $C^{1,0}_{1,0} = \{q_4\}$, $C^{1,0}_{1,1} = \{q_4, q_5\}$, $C^{1,0}_{2,0} = \{q_6\}$, $C^{1,0}_{2,1} = \{q_6, q_7\}$, $C^{1,1}_{0,0} = \{q_8\}$, $C^{1,1}_{0,1} = \{q_8, q_9\}$, $C^{1,1}_{1,0} = \{q_{10}\}$, $C^{1,1}_{1,1} = \{q_{10}, q_{11}\}$, $C^{1,1}_{2,0} = \{q_{12}\}$, $C^{1,1}_{2,1} = \{q_{12}, q_{13}\}$. Then the Muller automaton $\mathcal{A}$ contains a maximal alternating tree of length $\omega^1 \cdot 3 + \omega^0 \cdot 2$ given by the sequence $(C^{i,j}_{k,l})_{i \leq 1, j < 2, k < 3, l \leq 1}$.

At the beginning of the computation, an infinite input s (usually binary) is written on the input tape. In both cases, a computation of $\mathcal{M}$ on s is defined in the usual way. An infinite sequence of successive states visited by $\mathcal{M}$ during the processing of s is called an infinite run of $\mathcal{M}$ on s, denoted by ρ_s. The set of states appearing infinitely often in ρ_s is denoted by $\inf(\rho_s)$

If $\mathcal{M}$ is a deterministic Turing machine, an infinite input stream s is said to be *accepted* by $(\mathcal{M}, \mathcal{F})$ or by $(\mathcal{M}, \mathcal{T})$ if the unique infinite run ρ_s satisfies $\inf(\rho_s) \cap \mathcal{F} \neq \emptyset$ or $\inf(\rho_s) \in \mathcal{T}$, respectively; the infinite input s is said to be *rejected* otherwise. If $\mathcal{M}$ is nondeterministic, s is said to be *accepted* by $(\mathcal{M}, \mathcal{F})$ or $(\mathcal{M}, \mathcal{T})$ if there exists an infinite run ρ_s such that $\inf(\rho_s) \cap \mathcal{F} \neq \emptyset$ or $\inf(\rho_s) \in \mathcal{T}$, respectively; s is *rejected* otherwise. The set of all words accepted by $(\mathcal{M}, \mathcal{F})$ or $(\mathcal{M}, \mathcal{T})$ is the *ω-language recognized* by $(\mathcal{M}, \mathcal{F})$ or $(\mathcal{M}, \mathcal{T})$, respectively.

Every ω-language recognized by some deterministic Büchi or Muller Turing machine belongs to the topological class $\mathbf{\Pi_2^0}$ or $BC(\mathbf{\Pi_2^0})$, respectively [159, Corollaries 3.3 and 3.4].[2] However, a simple cardinality argument shows that not all $\mathbf{\Pi_2^0}$-set and not all $BC(\mathbf{\Pi_2^0})$-sets can be recognized by some deterministic Büchi or Muller Turing machine, respectively: indeed, there are $\aleph_0$ Muller Turing machines and $2^{\aleph_0}$ sets in $\mathbf{\Pi_2^0}$ or in $BC(\mathbf{\Pi_2^0})$. Besides, the nondeterministic Büchi and Muller Turing machines are equivalent and strictly more powerful than their deterministic counterparts. They recognize precisely the class of *effectively analytic* ω-languages, denoted by Σ_1^1 (lightface) [159, Theorem 3.5]. The relation $\Sigma_1^1 \subsetneq \mathbf{\Sigma_1^1}$ holds [87].

Finally, we recall that the Muller acceptance condition is the most powerful one amongst those usually investigated in ω-automata theory (i.e., Büchi, Rabin, Streett, parity) [159, Corollaries 3.4, 3.5 and Theorem 3.5].

7.3 Boolean Neural Networks

7.3.1 The Model

We focus on Boolean recurrent neural networks as presented in Chapter 4, yet working on infinite input streams.

Formally, a *Boolean recurrent neural network* (denoted as BRNN) contains M Boolean input cells $(u_i)_{i=1}^M$, N Boolean internal neurons $(x_i)_{i=1}^N$, and P Boolean output cells $(x_j)_{j=i_1}^{i_P}$ chosen among the N internal ones. The dynamics of the network is computed as usual: given the activation values of the input and internal neurons $(u_j)_{j=1}^M$ and $(x_j)_{j=1}^N$ at time t, the activation values of each internal neuron x_i is updated by the following equation:

$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij}(t) \cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t) \cdot u_j(t) + c_i(t)\right) \text{ for } i = 1, \ldots, N \qquad (7.1)$$

where $a_{ij}(t)$, $b_{ij}(t)$, and $c_i(t)$ are rational values corresponding to the weights of the synaptic connections and the bias of the network at time t, and θ is the Heaviside step function.

[2] $BC(\mathbf{\Pi_2^0})$ is the finite Boolean combinations of $\mathbf{\Pi_2^0}$-sets, i.e., the collection of sets obtained by finite unions, intersections and complementations of $\mathbf{\Pi_2^0}$-sets.

In the sequel, such a network will sometimes be denoted as a tuple

$$\mathcal{N} = (X, U, V, a, b, c)$$

where $X = \{x_i : 1 \leq i \leq N\}$, $U = \{u_i : 1 \leq i \leq M\}$, and $V \subseteq U$ are the sets of activation, input, and output cells, respectively, and $a \in \mathbb{Q}^{N \times N}$, $b \in \mathbb{Q}^{N \times M}$, and $c \in \mathbb{Q}^{N \times 1}$ are rational matrices corresponding to the weights and bias of the network.[3]

The dynamics of any BRNN $\mathcal{N}$ is therefore given by the function $f_{\mathcal{N}} : \mathbb{B}^M \times \mathbb{B}^N \to \mathbb{B}^N$ defined by

$$f_{\mathcal{N}}(\boldsymbol{u}(t), \boldsymbol{x}(t)) = \boldsymbol{x}(t+1)$$

where the components of $\boldsymbol{x}(t+1)$ are given by Equation (7.1).

In Section 5.2, we have recalled that Boolean neural networks disclose same computational capabilities as finite state automata [89, 118, 119]. Besides, it can be observed that rational-weighted and real-weighted Boolean neural networks are actually computationally equivalent.[4]

Consider some BRNN $\mathcal{N}$ provided with M Boolean input cells and N sigmoidal internal cells. Assuming the initial state of the network to be $\boldsymbol{x}(0) = \mathbf{0}$, any infinite input stream

$$s = (\boldsymbol{u}(t))_{t \in \mathbb{N}} = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2) \cdots \in (\mathbb{B}^M)^\omega$$

induces via Equation (7.1) an infinite sequence of consecutive states

$$c_s = (\boldsymbol{x}(t))_{t \in \mathbb{N}} = \boldsymbol{x}(0)\boldsymbol{x}(1) \cdots \in (\mathbb{B}^N)^\omega$$

called the *(Boolean) computation* of $\mathcal{N}$ induced by s. A computation of a BRNN is illustrated in Figure 26.

Note that any BRNN $\mathcal{N}$ with N internal cells can only have 2^P – i.e., finitely many – possible distinct states. Consequently, for any infinite computation c_s, there necessarily exists at least one state that recurs infinitely often in c_s. In fact, any computation c_s necessarily consists of a finite prefix of output states followed by an infinite suffix of output states that repeat infinitely often – yet not necessarily in a periodic manner. The non-empty set of all the output states that repeat infinitely often in c_s will be denoted by $\inf(c_s)$. A set of states of the form $\inf(c_s)$ will be called an *attractor* for $\mathcal{N}$. A precise definition can be given as follows:

Definition 24. Let $\mathcal{N}$ be some BRNN. A set $A = \{\boldsymbol{y_0}, \ldots, \boldsymbol{y_k}\} \subseteq \mathbb{B}^N$ is an *attractor* for $\mathcal{N}$ if there exists some infinite input stream s such that the corresponding computation c_s satisfies $\inf(c_s) = A$.

In words, an attractor of $\mathcal{N}$ is a set of states into which the computation of the network could become forever trapped – yet not necessarily in a periodic manner – for some infinite input stream s. An attractor of some BRNN is illustrated in Figure 26.

[3] The consideration of real synaptic weights would not change the results of this section.

[4] Indeed, in both rational- and real-weighted cases, the number of Boolean states of the network is finite, and hence, the network can be simulated by some finite state automaton.

In bio-inspired complex systems, the concept of an *attractor* has been shown to carry strong computational implications, by being associated with different functions, such as memory, motor behavior, and classification. According to Kauffman: "Because many complex systems harbour attractors to which the system settle down, the attractors literally are most of what the systems do" [86, p.191]. Alternative attractors are commonly interpreted as alternative memories [12, 13, 45, 49, 72, 101, 102, 146]. In addition, *attractor dynamics* or *quasi-attractor dynamics* have been associated to perceptions, thoughts and memories, and the *chaotic intinerancy* between those with sequences in thinking, speaking and writing [79, 84, 166, 167, 168].

The central hypothesis for brain attractors is that, once activated by appropriate activity, the neural network behaviour shall be maintained by continuous reentry of activity [12]. Attractors must be stable at short time scales. Besides, whenever the same information is repeatedly presented in a network, the same pattern of activity is evoked in a circuit of functionally interconnected neurons called "cell assembly" [57, 68]. The cell assemblies would thus elicit some repeated ordered and precise interspike interval relationships, referred to as *preferred firing sequences*, or *spatiotemporal patterns of discharges*, which recur above chance levels [2, 3, 4, 5, 80, 116, 137, 174, 175, 178, 180] (see also the survey by Villa [176] and the references therein). These considerations point out the strong correlation existing between *attractor dynamics* of neural networks and *spatiotemporal patterns of discharges*: the spatiotemporal patterns would be witnesses of an underlying attractor dynamics. Associations between attractor dynamics and repeating spatiotemporal firing patterns have been experimentally observed in simulations of nonlinear dynamical systems [14, 15] as well as in simulations of large scale neuronal networks [77, 78]. In our context, the attractor dynamics of the Boolean networks are, whenever periodic, the precise phenomena that underly the arising of spatiotemporal patterns of discharges. This feature is illustrated in Figure 26.

Spatiotemporal patterns have been detected without a specific association to a stimulus in large networks of spiking neural networks or during spontaneous activity in electrophysiological recordings[78, 180]. These may be viewed as *spurious patterns* generated by *spurious attractors*. On the other hand, several examples exist of spatiotemporal firing patterns in behaving animals, from rats to primates, where preferred firing sequences can be associated to specific types of stimuli or behaviours [5, 137, 148, 178]. These can be viewed as *meaningful patterns* associated with *meaningful attractors*. But the attractors' meaningfulness cannot be reduced to the detection of a behavioural correlate. It could for instance be correlated to the build-up of higher order dynamics, like chaotic itinerancy [79, 84, 157, 166, 167, 168].

In this work, we suppose that the attractors are of two distinct types: either *meaningful* or *spurious*. The type of each attractor could be determined by its neurophysiological significance with respect to measurable observations, e.g. associated with certain behaviors or sensory discriminations. The classification of these attractors into meaningful or spurious types is not the subject of this work. Hence, from now on, we assume that any BRNN is a priori equipped with a corresponding classification of all of its attractors into meaningful and spurious types. Further discussions about the type specification of the attractors will be addressed in the

forthcoming sections.

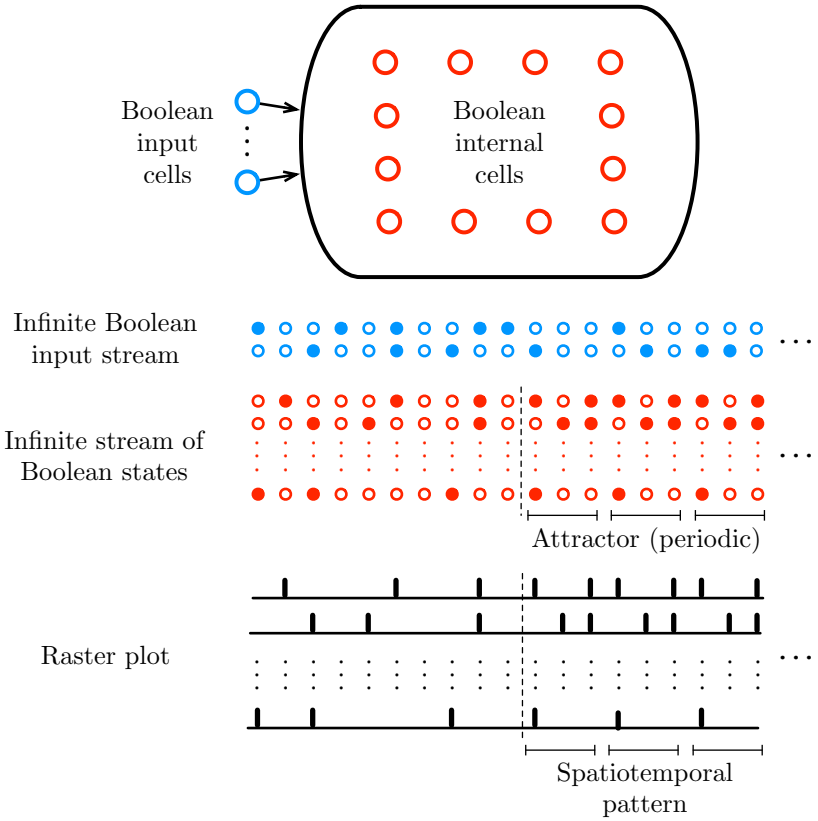


Figure 26 – Illustration of the computational process performed by some BRNN. The infinite Boolean input stream $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots \in (\mathbb{B}^M)^\omega$, represented by the blue pattern, induces a corresponding infinite stream of network's states – or computation – $c_s = \boldsymbol{x}(0)\boldsymbol{x}(1)\boldsymbol{x}(2)\cdots \in (\mathbb{B}^N)^\omega$, represented by the red pattern. The filled and empty circles represent active and quiet Boolean cells, respectively. From some time step onwards, a certain set states begins to repeat infinitely often: this corresponds the attractor associated to the input stream s. If the attractor occurs in a periodic manner, it corresponds to some spatiotemporal pattern, illustrated by the the raster plot.

According to these considerations, an infinite input stream s of $\mathcal{N}$ is called *meaningful* if $\inf(c_s)$ is a meaningful attractor, and it is called *spurious* if $\inf(c_s)$ is a spurious attractor. In other words, an input stream is called meaningful (respectively spurious) if the network dynamics induced by this input stream will eventually become confined into some meaningful (respectively spurious) attractor. Then, the set of all meaningful input streams of $\mathcal{N}$ is called the *neural language* of $\mathcal{N}$, denoted by $L(\mathcal{N})$. Finally, an arbitrary set of input streams $L \subseteq (\mathbb{B}^M)^\omega$ is said to be *recognizable* by some Boolean neural network if there exists a network $\mathcal{N}$ such that $L(\mathcal{N}) = L$.

Besides, if $\mathcal{N}$ denotes some Boolean neural network provided with an additional specification of the type of each of its attractors, then the *complementary* network $\mathcal{N}^{\complement}$ is defined to be the same network as $\mathcal{N}$ yet with a completely opposite type specification of its attractors. Then, an attractor A is meaningful for $\mathcal{N}^{\complement}$ iff

A is a spurious attractor for $\mathcal{N}$ and one has $L(\mathcal{N}^{\complement}) = L(\mathcal{N})^{\complement}$. All the preceding definitions are illustrated in the following Example 25.

Example 25. Consider the network $\mathcal{N}$ illustrated in Figure 8. Let us assume that the meaningful attractors of $\mathcal{N}$ are precisely those containing the state $(1,1,1)^T$, i.e., those involving the network's state where the three cells x_1, x_2, x_3 do simultaneously fire. All other attractors are assumed to be spurious.

Consider the periodic input stream $s = \left[\binom{0}{0}\binom{1}{0}\binom{0}{1}\right]^{\omega}$ and its corresponding computation

$$c_s = \binom{0}{0}{0}\left[\binom{0}{0}{0}\binom{1}{0}{0}\binom{0}{1}{1}\right]^{\omega}$$

From time step $t = 1$, the computation c_s of $\mathcal{N}$ remains confined in a cyclic visit of the states $\inf(c_s) = \{(0,0,0)^T, (1,0,0)^T, (0,1,1)^T\}$. Hence, this set is an attractor of $\mathcal{N}$. Since the state $(1,1,1)^T$ does not belong to this attractor, it is spurious. Therefore, the input stream s is also spurious, i.e. $s \notin L(\mathcal{N})$.

Consider the other periodic input stream $s' = \left[\binom{1}{1}\right]^{\omega}$ and its corresponding computation

$$c_{s'} = \left[\binom{0}{0}{0}\binom{1}{0}{0}\binom{1}{1}{1}\binom{0}{1}{1}\right]^{\omega}$$

The set of states $\inf(c_{s'}) = \{(0,0,0)^T, (1,0,0)^T, (1,1,1)^T, (0,1,1)^T\}$ is an attractor, and the computation $c_{s'}$ of $\mathcal{N}$ is confined in $\inf(c_{s'})$ already from the very first time step $t = 0$. In this case, the attractor is meaningful, since $(1,1,1)^T \in \inf(c_{s'})$, . It follows that the input stream s' is also meaningful, i.e., $s' \in L(\mathcal{N})$.

7.3.2 Expressive Power: Part 1

We provide a generalization to this precise attractor-based computational paradigm of the classical equivalence result between Boolean neural networks and finite state automata [89, 118, 119]. More precisely, we show that, under some natural specific conditions on the specification of the type of their attractors, Boolean recurrent neural networks disclose the very same expressive power as deterministic Büchi automata. Accordingly, the Wagner hierarchy of ω-regular languages [182] can be transposed from the Büchi automaton to the neural network context, and a hierarchical classification of Boolean neural networks can be deduced. The obtained classification is intimately related to the attractor properties of the neural networks, and hence provides a new refined measure of the computational power of Boolean neural networks in terms of their attractor dynamics.

Boolean Recurrent Neural Networks and Büchi Automata

We now prove that, under some natural conditions on the type specification of their attractors, Boolean recurrent neural networks are computationally equivalent to deterministic Büchi automata.

More precisely, we assume that the type specification of the attractors of a network $\mathcal{N}$ is naturally related to its output layer as follows: an attractor $A = \{\boldsymbol{y_0}, \ldots, \boldsymbol{y_k}\}$ of $\mathcal{N}$ is considered *meaningful* if it contains at least one state where some output cell is spiking, i.e. if there exist $i \leq k$ and $j \leq N$ such that $x_j \in V$ and $(\boldsymbol{y_i})_j = 1$;

the attractor A is considered *spurious* otherwise. According to these assumptions, meaningful attractors refer to the cyclic behaviours of the network that induce some response activity of the system via its output layer, whereas spurious attractors refer to the cyclic behaviours of the system that do not evoke any response at all of the output layer.

It can be stated that the expressive powers of Boolean recurrent neural networks and deterministic Büchi automaton are equivalent. As a first step towards this result, the following proposition shows that any Boolean recurrent neural network can be simulated by some deterministic Büchi automaton.

Proposition 26. *Let $\mathcal{N}$ be some Boolean recurrent neural network provided with an output layer. Then there exists a deterministic Büchi automaton $\mathcal{A}_\mathcal{N}$ such that $L(\mathcal{N}) = L(\mathcal{A}_\mathcal{N})$.*

Proof. Let $\mathcal{N}$ be some neural network given by the tuple (X, U, V, a, b, c), with $|X| = N$, $|U| = M$, and $V = \{x_{i_1}, \ldots, x_{i_{M'}}\} \subseteq X$. Consider the deterministic Büchi automaton $\mathcal{A}_\mathcal{N} = (Q, A, i, \delta, \mathcal{F})$, where $Q = \mathbb{B}^N$, $A = \mathbb{B}^M$, i is the N-dimensional zero vector, $\mathcal{F} = \{\boldsymbol{x} \in Q : (\boldsymbol{x})_{i_k} = 1 \text{ for some } 1 \leq k \leq M'\}$, and $\delta : Q \times A \to Q$ is the function defined by $\delta(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}'$ iff $\boldsymbol{x}' = \theta(a \cdot \boldsymbol{x} + b \cdot \boldsymbol{u} + \boldsymbol{c})$. Note that the complexity of the transformation is exponential, since $|Q| = 2^N$ and $|A| = 2^M$.

According to this construction, any infinite computation c_s of $\mathcal{N}$ naturally induces a corresponding infinite initial path $\rho(c_s)$ in $\mathcal{A}_\mathcal{N}$. Moreover, by the definitions of meaningful and spurious attractors of $\mathcal{N}$, an infinite input stream s is meaningful for $\mathcal{N}$ iff s is recognized by $\mathcal{A}_\mathcal{N}$. In other words, $s \in L(\mathcal{N})$ iff $s \in L(\mathcal{A}_\mathcal{N})$, and therefore $L(\mathcal{N}) = L(\mathcal{A}_\mathcal{N})$. □

According to the construction given in the proof of Proposition 26, any infinite computation of the network $\mathcal{N}$ is naturally associated with a corresponding infinite initial path in the automaton $\mathcal{A}_\mathcal{N}$, and conversely, any infinite initial path in $\mathcal{A}_\mathcal{N}$ corresponds to some possible infinite computation of $\mathcal{N}$. Consequently, there is a biunivocal correspondence between the *attractors* of the network $\mathcal{N}$ and the *cycles* in the graph of the corresponding Büchi automaton $\mathcal{A}_\mathcal{N}$. As a result, a procedure to compute all possible attractors of a given network $\mathcal{N}$ is obtained by firstly constructing the corresponding deterministic Büchi automaton $\mathcal{A}_\mathcal{N}$ and secondly listing all cycles in the graph of $\mathcal{A}_\mathcal{N}$.

Conversely, we prove now that any deterministic Büchi automaton can be simulated by some Boolean recurrent neural network.

Proposition 27. *Let $\mathcal{A}$ be some deterministic Büchi automaton over the alphabet $\mathbb{B}^M$, with $M \geq 1$. Then there exists a Boolean recurrent neural network $\mathcal{N}_\mathcal{A}$ provided with an output layer such that $L(\mathcal{A}) = L(\mathcal{N}_\mathcal{A})$.*

Proof. Let $\mathcal{A} = (Q, \mathbb{B}^M, q_1, \delta, \mathcal{F})$ be some deterministic Büchi automaton over alphabet $\mathbb{B}^M$, with $Q = \{q_1, \ldots, q_N\}$, and $\mathcal{F} = \{q_{i_1}, \ldots, q_{i_K}\} \subseteq Q$. Consider the network $\mathcal{N}_\mathcal{A} = (X, U, V, a, b, c)$ with $2^M + N + 1 + M$ cells given as follows: firstly, $X = \{x_i : 0 \leq i \leq 2^M + N\}$, where X is decomposed into a set of 2^M "letter cells" $X_L = \{x_i : 0 \leq i < 2^M\}$, a "delay-cell" x_{2^M}, and a set of N "state cells" $X_S = \{x_i : 2^M < i \leq 2^M + N\}$; secondly, the set of $|M|$ "input units" $U = \{u_0, \ldots, u_{M-1}\}$, and thirdly, the outptut layer $V = \{x_{2^M+j} : q_j \in \mathcal{F}\}$. The

idea of the simulation is that the "letter cells" and "state cells" of the network $\mathcal{N}_\mathcal{A}$ simulate the letters and states currently read and entered by the automaton $\mathcal{A}$, respectively.

Towards this purpose, the weight matrices a, b, and c are described as follows. Concerning the matrix b: for any $x_k \in X_L$, we consider the binary decomposition of k, namely $k = \sum_{j=0}^{M-1} \beta_{kj} \cdot 2^j$, with $\beta_{kj} \in \{0,1\}$, and for any $0 \leq j < M$, we set the weight $b_{k,j} = \beta_{kj} \cdot 2^j + (\beta_{kj} - 1)$; for all other k, we set $b_{k,j} = 0$, for any $0 \leq j < M$. Concerning the matrix c: for any $x_k \in X_L$, we set $c_k = 1 - k$; we also set $c_{2^M} = c_{2^M+1} = 1$; for all other k, we set $c_k = 0$. Concerning the matrix a: we set $a_{2^M+1,2^M} = -1$, and for any $x_k \in X_L$ and any $x_{2^M+i}, x_{2^M+j} \in X_S$, we set $a_{2^M+j,k} = a_{2^M+j,2^M+i} = 1/2$ iff $(q_i, \boldsymbol{\beta}_k, q_j)$ is a transition of $\mathcal{A}$; otherwise, for any pair of indices $i_1, i_2 \in \{0, \ldots, 2^M + N\}$ such that a_{i_1,i_2} has not been set to -1 or $1/2$, we set $a_{i_1,i_2} = 0$. This construction is illustrated in Figure 27.

According to this construction, if we let $\boldsymbol{\beta}_k$ denote the boolean vector whose components are the β_{kj}'s (for $0 \leq j < M$), one has that the "letter cell" x_k will spike at time $t+1$ iff the input vector $\boldsymbol{\beta}_k \in \mathbb{B}^M$ is received at time t. Moreover, at every time step $t > 0$, a unique "letter cell" $x_k \in X_L$ and "state cell" $x_{2^M+i} \in X_S$ are spiking, and, if $\mathcal{A}$ performs the transition $(q_i, \boldsymbol{\beta}_k, q_j)$ at time t, then network $\mathcal{N}_\mathcal{A}$ evokes the spiking pattern $x_k(t) = x_{2^M+i}(t) = x_{2^M+j}(t+1) = 1$. The relation between the final states $\mathcal{F}$ of $\mathcal{A}$ and the output layer V of $\mathcal{N}_\mathcal{A}$ ensures that any infinite input stream $s \in (\mathbb{B}^M)^\omega$ is recognized by $\mathcal{A}$ if and only if s is meaningful for $\mathcal{N}_\mathcal{A}$. Therefore, $L(\mathcal{A}) = L(\mathcal{N}_\mathcal{A})$. □

Propositions 26 and 27 yield to the following equivalence between recurrent neural networks and deterministic Büchi automata.

Theorem 28. *Let $L \subseteq (\mathbb{B}^k)^\omega$ for some $k \geq 1$. Then L is recognizable by some Boolean recurrent neural network provided with an output layer iff L is recognizable by some deterministic Büchi automaton.*

Proof. Proposition 26 shows that every language recognizable by some Boolean recurrent neural network is also recognizable by some deterministic Büchi automaton. Conversely, Proposition 27 shows that every language recognizable by some deterministic Büchi automaton is also recognizable by some Boolean recurrent neural network. □

The two procedures given in the proofs of Propositions 26 and 27 are illustrated in the following Example 29.

Example 29. Consider the network $\mathcal{N}$ described in Example 3, and suppose that the output layer of $\mathcal{N}$ consists of the unique cell x_3, i.e. $V = \{x_3\}$. The translation from this network $\mathcal{N}$ to its corresponding deterministic Büchi automaton $\mathcal{A}_\mathcal{N}$ is illustrated in Figure 28, panels a and b. Proposition 26 ensures that $L(\mathcal{N}) = L(\mathcal{A}_\mathcal{N})$. Conversely, the translation from a simple deterministic Büchi automaton $\mathcal{A}$ to its corresponding neural network $\mathcal{N}_\mathcal{A}$ is illustrated in Figures 28, panels c and d. Proposition 27 ensures that $L(\mathcal{A}) = L(\mathcal{N}_\mathcal{A})$.

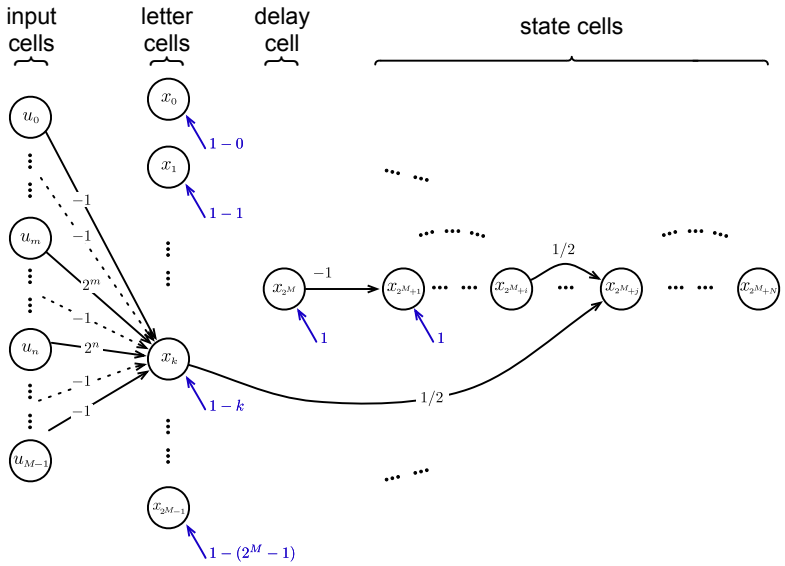


Figure 27 – The network $\mathcal{N}_\mathcal{A}$ described in the proof of Proposition 27. $\mathcal{N}_\mathcal{A}$ is characterized by a set of M input cells $U = \{u_0, \ldots, u_{M-1}\}$ reading the alphabet $\mathbb{B}^M$, 2^M "letter cells" $X_L = \{x_i : 0 \leq i < 2^M\}$, a "delay-cell" x_{2^M}, and a set of N "state cells" $X_S = \{x_i : 2^M < i \leq 2^M + N\}$. The idea of the simulation is that the "letter cells" and "state cells" of the network $\mathcal{N}_\mathcal{A}$ simulate the letters and states currently read and entered by the automaton $\mathcal{A}$, respectively. In this illustration, we assume that the binary decomposition of k is given by $k = 2^m + 2^n$, so that the "letter cell" x_k receives synaptic connections of intensities 2^m and 2^n from input cells u_m and u_n, respectively, and it receives synaptic connections of intensities -1 from any other input cells. Consequently, the "letter cell" x_k becomes active at time $t+1$ iff the sole input cells u_m and u_n are active at time t. The synaptic connections to other "letter cells" are not illustrated. Moreover, the synaptic connections $a_{2^M+j,k} = a_{2^M+j,2^M+i} = 1/2$ model the transition (q_i, β_k, q_j) of automaton $\mathcal{A}$. The synaptic connections modelling other transitions are not illustrated.

THE BRNN HIERARCHY

In the theory of infinite word reading machines, abstract devices are commonly classified according to the topological complexity of their underlying ω-language (i.e., the languages of infinite words that they recognize). Such classifications provide an interesting measure of the expressive power of various kinds of infinite word reading machines. In this context, the most refined hierarchical classification of ω-automata – or equivalently, of ω-rational languages – is the so-called *Wagner hierarchy* [182].

According to Theorem 28, the Wagner hierarchy can naturally be transposed from Büchi automata to Boolean neural networks. As a result, a hierarchical classification of first-order Boolean recurrent neural networks is obtained. The hierarchical classification naturally induces a measure of complexity for Boolean neural networks based on their attractor dynamics. Notably, this complexity refers to the ability of the networks to perform more or less complicated classification tasks of their inputs via the manifestation of meaningful or spurious attractor dynamics.

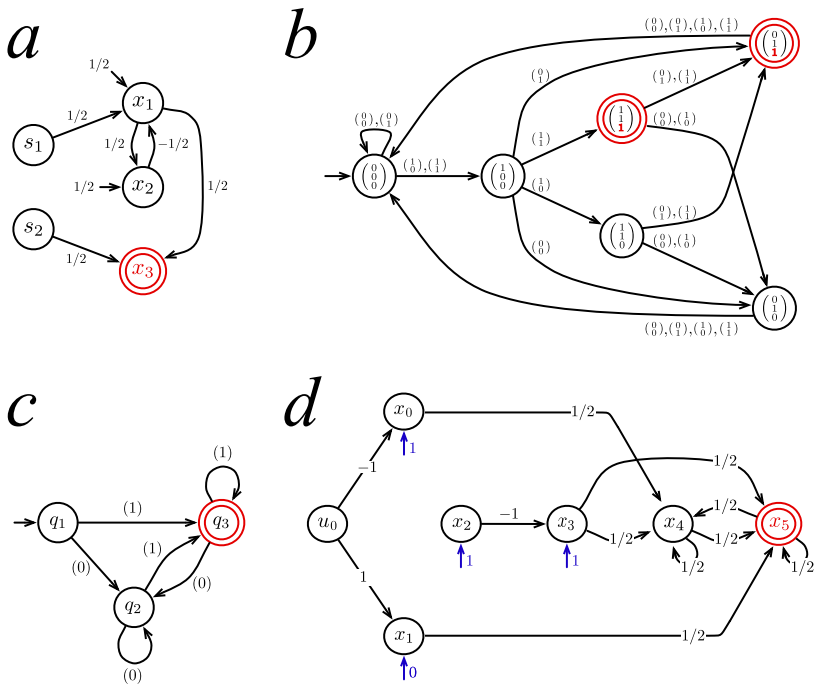


Figure 28 – Panels a, b. Translation from a neural network to its corresponding deterministic Büchi automaton. **a.** The neural network $\mathcal{N}$ of Figure 8 provided with an additional specification of an output layer $V = \{x_3\}$, denoted in red. **b.** The deterministic Büchi automaton $\mathcal{A}_\mathcal{N}$ corresponding to the neural network $\mathcal{N}$. According to the proof of Proposition 26, the nodes of $\mathcal{A}_\mathcal{N}$ are the states (i.e., the spiking configurations) of $\mathcal{N}$, and there is an edge labelled by $\boldsymbol{a}$ from $\boldsymbol{q}$ to $\boldsymbol{q'}$ in $\mathcal{A}_\mathcal{N}$ iff $\mathcal{N}$ switches from state $\boldsymbol{q}$ to $\boldsymbol{q'}$ when it receives input $\boldsymbol{a}$. The final states of $\mathcal{A}_\mathcal{N}$ are denoted in red.
Panels c, d. Translation from a deterministic Büchi automaton to its corresponding neural network. **c.** A deterministic Büchi automaton $\mathcal{A}$ with three states. The initial state q_1 is denoted with an incoming edge, and the final state q_3 is emphasized in red. **d.** The network $\mathcal{N}_\mathcal{A}$ corresponding to the Büchi automaton $\mathcal{A}$. According to the proof of Proposition 27, the network $\mathcal{N}_\mathcal{A}$ contains one ($M = 1$) input neuron u_0, two ($2^M = 2$) "letter cells" x_0 and x_1, one "delay cell" x_2, and three ($N = 3$) "state cells" x_3, x_4, x_5 each one being associated with a corresponding state of the automaton. The output layer is represented by the cell x_5, denoted in red and double-circled. The background activities are labeled in blue.

Before the result is stated, the following definitions need to be introduced. Given two Boolean recurrent neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$ with M_1 and M_2 input units respectively, we say that $\mathcal{N}_1$ *reduces* (or *Wadge reduces* or *continuously reduces*) to $\mathcal{N}_2$, denoted by $\mathcal{N}_1 \leq_W \mathcal{N}_2$, iff there exists a continuous function $f : (\mathbb{B}^{M_1})^\omega \to (\mathbb{B}^{M_2})^\omega$ such that, for any input stream $s \in (\mathbb{B}^{M_1})^\omega$, one has $s \in L(\mathcal{N}_1) \Leftrightarrow f(s) \in L(\mathcal{N}_2)$, or equivalently, such that $L(\mathcal{N}_1) = f^{-1}(L(\mathcal{N}_2))$ [181]. Intuitively, $\mathcal{N}_1 \leq_W \mathcal{N}_2$ iff the problem of determining whether some input stream s belongs to the neural language of $\mathcal{N}_1$ (i.e. whether s is meaningful for $\mathcal{N}_1$) reduces via some simple function f to the problem of knowing whether $f(s)$ belongs to the neural language of $\mathcal{N}_2$ (i.e. whether s is meaningful for $\mathcal{N}_2$). The corresponding strict reduction,

equivalence relation, and incomparability relation are then naturally defined by $\mathcal{N}_1 <_W \mathcal{N}_2$ iff $\mathcal{N}_1 \leq_W \mathcal{N}_2 \not\leq_W \mathcal{N}_1$, as well as $\mathcal{N}_1 \equiv_W \mathcal{N}_2$ iff $\mathcal{N}_1 \leq_W \mathcal{N}_2 \leq_W \mathcal{N}_1$, and $\mathcal{N}_1 \perp_W \mathcal{N}_2$ iff $\mathcal{N}_1 \not\leq_W \mathcal{N}_2 \not\leq_W \mathcal{N}_1$. Moreover, a network $\mathcal{N}$ is called *self-dual* if $\mathcal{N} \equiv_W \mathcal{N}^{\complement}$; it is called *non-self-dual* if $\mathcal{N} \not\equiv_W \mathcal{N}^{\complement}$, which can be proved to be equivalent to saying that $\mathcal{N} \perp_W \mathcal{N}^{\complement}$ [181]. We recall that the network $\mathcal{N}^{\complement}$ corresponds to the network $\mathcal{N}$ whose type specification of its attractors has been inverted. Consequently, $\mathcal{N}^{\complement}$ does not correspond *a priori* to some neural network provided with an output layer. By extension, an $\equiv_W$-equivalence class of networks is called *self-dual* if all its elements are self-dual, and *non-self-dual* if all its elements are non-self-dual.

The continuous reduction relation over the class of Boolean recurrent neural networks naturally induces a hierarchical classification of networks formally defined as follows:

Definition 30. The collection of all Boolean recurrent neural networks ordered by the reduction "$\leq_W$" is called the *BRNN hierarchy*.

We now turn to the characterization of the BRNN hierarchy. For this purpose, let us define the *DBA hierarchy* to be the collection of all deterministic Büchi automata over multidimensional Boolean alphabets $\mathbb{B}^k$ ordered by the continuous reduction relation "$\leq_W$". More precisely, given two deterministic Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$, we set $\mathcal{A}_1 \leq_W \mathcal{A}_2$ iff there exists a continuous function f such that, for any input stream s, one has $s \in L(\mathcal{A}_1) \Leftrightarrow f(s) \in L(\mathcal{A}_2)$. The following result shows that the BRNN hierarchy and the DBA hierarchy are actually isomorphic. Moreover, a possible isomorphism is given by the mapping described in Proposition 26 which associates to every network $\mathcal{N}$ a corresponding deterministic Büchi automaton $\mathcal{A}_{\mathcal{N}}$.

Proposition 31. *The BRNN hierarchy and the DBA hierarchy are isomorphic.*

Proof. Consider the mapping described in Proposition 26 which associates to every network $\mathcal{N}$ a corresponding deterministic automaton $\mathcal{A}_{\mathcal{N}}$. We prove that this mapping is an embedding from the BRNN hierarchy into the DBA hierarchy. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be any two networks, and let $\mathcal{A}_{\mathcal{N}_1}$ and $\mathcal{A}_{\mathcal{N}_2}$ be their corresponding deterministic Büchi automata. Proposition 26 ensures that $L(\mathcal{N}_1) = L(\mathcal{A}_{\mathcal{N}_1})$ and $L(\mathcal{N}_2) = L(\mathcal{A}_{\mathcal{N}_2})$. Hence, one has $\mathcal{N}_1 \leq_W \mathcal{N}_2$ iff $\mathcal{A}_{\mathcal{N}_1} \leq_W \mathcal{A}_{\mathcal{N}_2}$, and thus $\mathcal{N}_1 <_W \mathcal{N}_2$ iff $\mathcal{A}_{\mathcal{N}_1} <_W \mathcal{A}_{\mathcal{N}_2}$, which shows that the considered mapping is an embedding. We now show that, up to the continuous equivalence "$\equiv_W$", this mapping is also onto. Let $\mathcal{A}$ be some deterministic Büchi automaton. By Proposition 27, there exists a network $\mathcal{M} = \mathcal{N}_{\mathcal{A}}$ such that $L(\mathcal{A}) = L(\mathcal{M})$. By Proposition 26, there exists an automaton $\mathcal{A}_{\mathcal{M}}$ such that $L(\mathcal{A}_{\mathcal{M}}) = L(\mathcal{M}) = L(\mathcal{A})$. Hence, $\mathcal{A}_{\mathcal{M}} \equiv_W \mathcal{A}$. Therefore, for any deterministic Büchi automaton $\mathcal{A}$, there exists a neural network $\mathcal{M}$ such that $\mathcal{A}_{\mathcal{M}} \equiv_W \mathcal{A}$, showing that up to the continuous equivalence relation "$\equiv_W$", the mapping $\mathcal{N} \mapsto \mathcal{A}_{\mathcal{N}}$ is onto. □

By Proposition 31 and the usual results concerning the DBA hierarchy, a precise description of the BRNN hierarchy can be given. In fact, the BRNN hierarchy is well-founded, i.e. there is no infinite strictly descending sequence of networks

$\mathcal{N}_0 >_W \mathcal{N}_1 >_W \mathcal{N}_2 >_W \dots$. The maximal strict chains[5] and antichains[6] of the BRNN hierarchy have length $\omega + 1$ and 2, respectively, meaning that the BRNN hierarchy has a height of $\omega + 1$ and a width of 2. It can also be shown that incomparable networks are equivalent (for the relation $\equiv_W$) up to complementation, i.e., for any two networks $\mathcal{N}_1$ and $\mathcal{N}_2$, one has $\mathcal{N}_1 \perp_W \mathcal{N}_2$ iff $\mathcal{N}_1$ and $\mathcal{N}_2$ are non-self-dual and $\mathcal{N}_1 \equiv_W \mathcal{N}_2^{\complement}$. Consequently, up to equivalence and complementation, the BRNN hierarchy is actually a well-ordering. In fact, the BRNN hierarchy consists of an infinite alternating succession of pairs of non-self-dual and single self-dual classes, overhung by an additional single non-self-dual class at the first limit level ω, as illustrated in Figure 29.

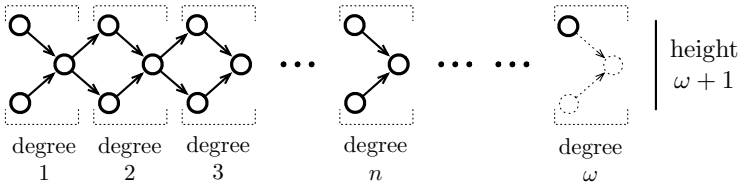


Figure 29 – The BRNN hierarchy: an infinite alternating succession of pairs of non-self-dual classes of networks followed by single self-dual classes of networks, all of them overhung by an additional single non-self-dual class at the first limit level. Circles represent the equivalence classes of networks (with respect to the relation "$\equiv_W$") and arrows between circles represent the strict reduction "$<_W$" between all elements of the corresponding classes.

For convenience reasons, the degree of a network $\mathcal{N}$ in the BRNN hierarchy is defined such that the same degree is shared by both non-self-dual networks at some level and self-dual networks located just one level higher, as illustrated in Figure 29:

$$d(\mathcal{N}) = \begin{cases} 1 & \text{if } L(\mathcal{N}) = \emptyset \text{ or } \emptyset^{\complement}, \\ \sup\{d(\mathcal{M}) + 1 : \mathcal{M} \text{ non-self-dual and } \mathcal{M} <_W \mathcal{N}\} & \text{if } \mathcal{N} \text{ is non-self-dual}, \\ \sup\{d(\mathcal{M}) : \mathcal{M} \text{ non-self-dual and } \mathcal{M} <_W \mathcal{N}\} & \text{if } \mathcal{N} \text{ is self-dual}. \end{cases}$$

Furthermore, the equivalence between the DBA and BRNN hierarchies implies that the BRNN hierarchy is *decidable*, i.e., that there exists an algorithmic procedure to compute the degree of any network. All the above properties of the BRNN hierarchy are summarized in the following result.

Theorem 32. *The BRNN hierarchy is a decidable pre-well-ordering of width* 2 *and height* $\omega + 1$.

Proof. The DBA hierarchy consists of a decidable pre-well-ordering of width 2 and height $\omega + 1$ [134]. By Proposition 31, the BRNN hierarchy and the DBA hierarchy are isomorphic. □

[5] A strict chain of length α in the BRNN hierarchy is a sequence of neural networks $(\mathcal{N}_k)_{k\in\alpha}$ such that $\mathcal{N}_i <_W \mathcal{N}_j$ iff $i < j$; a strict chain is said to be maximal if its length is at least as large as the length of every other strict chain.

[6] An antichain of length α in the BRNN hierarchy is a sequence of neural networks $(\mathcal{N}_k)_{k\in\alpha}$ such that $\mathcal{N}_i \perp_W \mathcal{N}_j$ for all $i, j \in \alpha$ with $i \neq j$; an antichain is said to be maximal if its length is at least as large as the length of every other antichain.

The following result provides a detailed description of the decidability procedure of the BRNN hierarchy. It is shown that the degree of a network $\mathcal{N}$ in the BRNN hierarchy is linked to the intricacy of its set of attractors, and more precisely, corresponds to the maximal number of times that this network could alternate between meaningful and spurious attractors along some computation.

Theorem 33. *Let $\mathcal{N}$ be some network provided with an additional specification of an output layer, $\mathcal{A}_\mathcal{N}$ be the corresponding deterministic Büchi automaton of $\mathcal{N}$, and $n > 0$.*

- *If there exists in $\mathcal{A}_\mathcal{N}$ a maximal alternating chain of length n and no maximal co-alternating chain of length n, then $d(\mathcal{N}) = n$ and $\mathcal{N}$ is non-self-dual.*
- *Symmetrically, if there exists in $\mathcal{A}_\mathcal{N}$ a maximal co-alternating chain of length n but no maximal alternating chain of length n, then also $d(\mathcal{N}) = n$ and $\mathcal{N}$ is non-self-dual.*
- *If there exist in $\mathcal{A}_\mathcal{N}$ a maximal alternating chain of length n as well as a maximal co-alternating chain of length n, then $d(\mathcal{N}) = n$ and $\mathcal{N}$ is self-dual.*
- *If there exist in $\mathcal{A}_\mathcal{N}$ a maximal alternating chain of length ω, then $d(\mathcal{N}) = \omega$ and $\mathcal{N}$ is non-self-dual.*

Proof. By Proposition 31, the degree of a network $\mathcal{N}$ in the BRNN hierarchy is equal to the degree of its corresponding deterministic Büchi automaton $\mathcal{A}_\mathcal{N}$ in the DBA hierarchy. Moreover, the degree of a deterministic Büchi automaton in the DBA hierarchy corresponds precisely to the length of a maximal alternating or co-alternating chain contained in this automaton [134]. □

By Theorem 33, the decidability procedure of the degree of a neural network $\mathcal{N}$ in the BRNN hierarchy consists firstly in translating the network $\mathcal{N}$ into its corresponding deterministic Büchi automaton $\mathcal{A}_\mathcal{N}$, as described in Proposition 26, and secondly in returning the ordinal $\alpha < \omega + 1$ corresponding to the length of a maximal alternating chain or co-alternating chain contained in $\mathcal{A}_\mathcal{N}$. Note that this procedure can clearly be achieved by some graph analysis of the automaton $\mathcal{A}_\mathcal{N}$, since the latter is always finite. Furthermore, since alternating and co-alternating chains are defined in terms of cycles in the graph of the automaton, and since cycles of $\mathcal{A}_\mathcal{N}$ do biunivocally correspond to attractors of $\mathcal{N}$, it can be deduced that the complexity of a network in the BRNN hierarchy is in fact directly related to the attractor dynamics of this network.

More precisely, the novel complexity measure induced by the BRNN hierarchy corresponds to the maximal number of times that the Boolean networks might alternate between meaningful and spurious attractors along their possible computations. Indeed, suppose that $d(\mathcal{N}) = n$. By Theorem 33, there necessarily exists some maximal alternating or co-alternating chain $(c_0, \dots, c_n)$ of length n in $\mathcal{A}_\mathcal{N}$. This means that every infinite initial path in $\mathcal{A}_\mathcal{N}$ might alternate at most n times between successful and non-successful cycles. Hence equivalently, every computation of $\mathcal{N}$ can only alternate at most n times between meaningful and spurious attractors before becoming eventually forever trapped by a last attractor. Now, suppose that $d(\mathcal{N}) = \omega$. By Theorem 33, there necessarily exists some maximal alternating chain (c_1, c_2) of length ω in $\mathcal{A}_\mathcal{N}$. Yet this corresponds to the existence

of an infinite initial path in $\mathcal{A}_{\mathcal{N}}$ that might alternate infinitely many times between cycles c_1 and c_2. Consequently, there also exists some computation of $\mathcal{N}$ that might alternate infinitely many times between some meaningful and spurious attractors.

Finally, the decidability procedure of the BRNN hierarchy is illustrated in Example 34 below.

Example 34. Let $\mathcal{N}$ be the neural network illustrated in Figure 28(*a*). Then $\mathcal{N}$ has degree ω in the BRNN hierarchy. Indeed, the corresponding deterministic Büchi automaton $\mathcal{A}_{\mathcal{N}}$ depicted in Figure 28(*b*) contains the two cycles

$$c_1 = \{(0,0,0)^T, (1,0,0)^T, (0,1,1)^T\} \text{ and } c_2 = \{(0,0,0)^T, (1,0,0)^T, (0,1,0)^T\}.$$

Cycle c_1 is successful, since it contains the final state $(0,1,1)^T$; cycle c_2 is not successful, since it contains no final state; and both cycles c_1 and c_2 are accessible from each other. Hence, the pair (c_1, c_2) is an alternating sequence of length ω contained in $\mathcal{A}_{\mathcal{N}}$. By Theorem 33, $d(\mathcal{N}) = \omega$ and $\mathcal{N}$ is non-self-dual. Observe that the input stream

$$s = \left[\binom{1}{1}\binom{0}{1}\binom{1}{1}\binom{1}{1}\binom{0}{0}\binom{1}{1}\right]^{\omega}$$

induces a corresponding computation c_s of $\mathcal{N}$ that alternates ω times between attractors c_1 and c_2.

7.3.3 Expressive Power: Part 2

We now show that by totally relaxing the restrictions on the type specification of their attractors, the networks significantly increase their expressive power from deterministic Büchi up to Muller automata. Hence, by transposing once again the Wagner classification theory from the Muller automaton to the neural network context, another refined hierarchical classification of Boolean neural networks is obtained. The classification induces a novel refined measure of complexity of Boolean recurrent neural networks based on their attractor dynamics.

Boolean Recurrent Neural Networks and Muller Automata

The assumption that the networks are provided with an additional description of an output layer, which would subsequently influence the type specification (meaningful/spurious) of their attractors, is now totally relaxed. Instead, we assume any network to be a priori provided with a precise classification of its attractors into meaningful and spurious types. The issue of the precise assignment of these attractors' types not considered here. The attractors' types could for instance be determined by their neurophysiological significance with respect to measurable observations associated to certain behaviours or sensory discriminations. Formally, we now assume that a recurrent neural network consists of a tuple $\mathcal{N} = (X, U, a, b, c)$ (the output layer V is not anymore specified) such that $\mathcal{N}$ is provided with an additional specification of each of its attractors into meaningful and spurious type.

In this context, we prove that the Boolean neural networks significantly increase their expressive powers from deterministic Büchi to Muller automata. The following straightforward generalization of Proposition 26 states that any such Boolean

recurrent neural network can be simulated by some deterministic Muller automaton.

Proposition 35. *Let $\mathcal{N}$ be some Boolean recurrent neural network provided with a type specification of each of its attractors. Then there exists a deterministic Muller automaton $\mathcal{A}_\mathcal{N}$ such that $L(\mathcal{N}) = L(\mathcal{A}_\mathcal{N})$.*

Proof. Let $\mathcal{N}$ be given by the tuple (X, U, a, b, c), with $|X| = N$, $|U| = M$, and let the meaningful attractors of $\mathcal{N}$ be given by $A_1, \ldots, A_K$, all others being spurious. Now, consider the deterministic Muller automaton $\mathcal{A}_\mathcal{N} = (Q, A, i, \delta, \mathcal{T})$, where $Q = \mathbb{B}^N$, $A = \mathbb{B}^M$, i is the N-dimensional zero vector, $\delta : Q \times A \to Q$ is defined by $\delta(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x'}$ iff $\boldsymbol{x'} = \theta\,(a \cdot \boldsymbol{x} + b \cdot \boldsymbol{u} + c)$, and $\mathcal{T} = \{A_1, \ldots, A_K\}$. According to this construction, any input stream s is meaningful for $\mathcal{N}$ iff s is recognized by $\mathcal{A}_\mathcal{N}$. In other words, $s \in L(\mathcal{N})$ iff $s \in L(\mathcal{A}_\mathcal{N})$, and therefore $L(\mathcal{N}) = L(\mathcal{A}_\mathcal{N})$. □

Conversely, as a generalization of Proposition 27, we show that any deterministic Muller automaton can be simulated by some Boolean recurrent neural network provided with a suitable type specification of its attractors.

Proposition 36. *Let $M > 0$ and let $\mathcal{A}$ be some deterministic Muller automaton over the alphabet $\mathbb{B}^M$. Then there exists a Boolean recurrent neural network $\mathcal{N}_\mathcal{A}$ provided with a type specification of each of its attractors such that $L(\mathcal{A}) = L(\mathcal{N}_\mathcal{A})$.*

Proof. Let $\mathcal{A}$ be given by the tuple $(Q, A, q_1, \delta, \mathcal{T})$, with $A = \mathbb{B}^M$, $Q = \{q_1, \ldots, q_N\}$ and $\mathcal{T} \subseteq \mathcal{P}(Q)$. Now, consider the network $\mathcal{N}_\mathcal{A} = (X, U, a, b, c)$ described in the proof of Proposition 27. The meaningful and spurious attractors of $\mathcal{N}_\mathcal{A}$ remain to be defined. As mentioned in the proof of Proposition 27, at every time step $t > 0$, only one among the "state cells" $\{x_{2^M+1}, \ldots, x_{2^M+N}\}$ is spiking. Hence, for any state $\boldsymbol{y}$ of $\mathcal{N}_\mathcal{A}$ that might occur at some time step $t > 0$, let $i(\boldsymbol{y}) \in \{1, \ldots, N\}$ be the index such that $x_{2^M+i(\boldsymbol{y})}$ is the unique "state cell" which is spiking during state $\boldsymbol{y}$. An attractor $\{\boldsymbol{y_0}, \ldots, \boldsymbol{y_k}\}$ of $\mathcal{N}_\mathcal{A}$ is then said to be meaningful iff $\{q_{i(\boldsymbol{y_0})}, \ldots, q_{i(\boldsymbol{y_k})}\} \in \mathcal{T}$.

Consequently, for any infinite infinite sequence $s \in (\mathbb{B}^M)^\omega$, the infinite path ρ_s in $\mathcal{A}$ satisfies $\inf(\rho_s) \in \mathcal{T}$ iff the computation c_s in $\mathcal{N}_\mathcal{A}$ is such that $\inf(c_s)$ is a meaningful attractor. Therefore, s is recognized by $\mathcal{A}$ iff s is meaningful for $\mathcal{N}_\mathcal{A}$, showing that $L(\mathcal{A}) = L(\mathcal{N}_\mathcal{A})$. □

Propositions 35 and 36 yield the following equivalence between Boolean recurrent neural networks and deterministic Muller automata.

Theorem 37. *Let $L \subseteq (\mathbb{B}^k)^\omega$ for some $k > 0$. Then the following conditions are equivalent:*

1. *L is recognizable by some Boolean recurrent neural network provided with a type specification of its attractors;*
2. *L is recognizable by some deterministic Muller automaton;*
3. *L is ω-rational.*

Proof. The equivalence between conditions 1 and 2 is given by Propositions 35 and 36. The equivalence between conditions 2 and 3 is a well-known result of automata theory [134]. □

The two procedures described in the proofs of Propositions 35 and 36 are illustrated in the following Example 38.

Example 38. Consider the network $\mathcal{N}$ described in Example 3 and assume the set of meaningful and spurious attractors of $\mathcal{N}$ has been established by some criterion. More precisely, assume that the sole meaningful attractor of $\mathcal{N}$ is $A = \{(0,0,0)^T, (1,0,0)^T, (0,1,1)^T\}$, all other ones being considered as spurious. The translation from the network $\mathcal{N}$ to its corresponding deterministic Muller automaton $\mathcal{A}_\mathcal{N}$ is illustrated in Figure 30, panels *a* and *b*. Proposition 35 ensures that $L(\mathcal{N}) = L(\mathcal{A}_\mathcal{N})$. Conversely, the translation from a simple deterministic Muller automaton $\mathcal{A}$ over the alphabet $\mathbb{B}^1$ to its corresponding network $\mathcal{N}_\mathcal{A}$ is illustrated in Figure 30, panels *c* and *d*. Proposition 36 ensures that $L(\mathcal{A}) = L(\mathcal{N}_\mathcal{A})$.

The Complete BRNN Hierarchy

We show that the collection of Boolean recurrent neural networks ordered by the continuous reduction corresponds to a refined hierarchical classification of height ω^ω. This classification induces a new refined measure of complexity for Boolean neural networks according based on their attractor dynamics. This hierarchical classification is formally defined as follows.

Definition 39. The collection of all Boolean recurrent neural networks provided with a type specification of their attractors ordered by the continuous reduction "$\leq_W$" is called the *complete BRNN hierarchy*.

We now turn to the characterization of the complete BRNN hierarchy. Towards this purpose, we recall that the collection of all deterministic Muller automata (over multidimensional Boolean alphabets $\mathbb{B}^k$) ordered by the continuous reduction "$\leq_W$" is commonly referred to as the *Wagner hierarchy* [182]. A generalization of Proposition 31 shows that the complete BRNN hierarchy and the Wagner hierarchy are isomorphic. A possible isomorphism is also given by the mapping described in Proposition 35 which associates to every network $\mathcal{N}$ a corresponding deterministic Muller automaton $\mathcal{A}_\mathcal{N}$.

Proposition 40. *The complete BRNN hierarchy and the Wagner hierarchy are isomorphic.*

Proof. Consider the mapping described in Proposition 35 which associates to every network $\mathcal{N}$ a corresponding deterministic Muller automaton $\mathcal{A}_\mathcal{N}$. A similar reasoning as the one presented in the proof of Proposition 31 shows that this mapping is an isomorphism between the complete BRNN hierarchy and the Wagner hierarchy. □

By Proposition 40 and the usual results on the Wagner hierarchy [182], the following description of the complete BRNN hierarchy can be given. The complete BRNN hierarchy also consists of a pre-well ordering of width 2, and any two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ satisfy the incomparability relation $\mathcal{N}_1 \perp_W \mathcal{N}_2$ iff $\mathcal{N}_1$ and $\mathcal{N}_2$ are

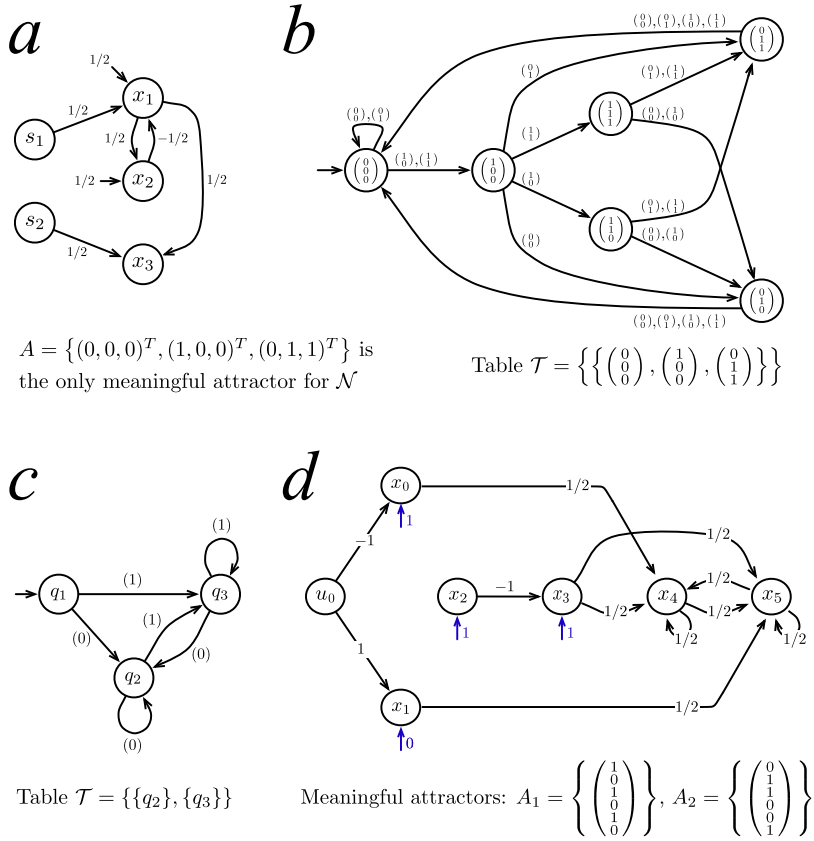


Figure 30 – Panels a, b. Translation from a Boolean neural network provided with a type specification of its attractors to its corresponding deterministic Muller automaton. **a.** A neural network $\mathcal{N}$ provided with an additional type specification of each of its attractors. In this case, $\mathcal{N}$ contains only one meaningful attractor determined by the following set of states $\{(0,0,0)^T, (1,0,0)^T, (0,1,1)^T\}$; all other ones are considered as spurious. **b.** The deterministic Muller automaton $\mathcal{A}_\mathcal{N}$ corresponding to the neural network $\mathcal{N}$ of panel *a*. According to the proof of Proposition 35, $\mathcal{A}_\mathcal{N}$ works over alphabet $\mathbb{B}^2$, contains six states, and possesses in its table $\mathcal{T}$ the sole cycle $\{(0,0,0)^T, (1,0,0)^T, (0,1,1)^T\}$, which corresponds to the sole meaningful attractor of $\mathcal{N}$.

Panels c, d. Translation from a deterministic Muller automaton to its corresponding Boolean neural network provided with a type specification of its attractors. **c.** A deterministic Muller automaton $\mathcal{A}$. The automaton works over alphabet $\mathbb{B}^1$, has three states, and possesses the two successful cycles $\{q_2\}$ and $\{q_3\}$, as mentioned by its table $\mathcal{T} = \{\{q_2\}, \{q_3\}\}$. **d.** The neural network $\mathcal{N}_\mathcal{A}$ corresponding to the Muller automaton $\mathcal{A}$ of panel *c*. According to the proof of Proposition 36, $\mathcal{N}_\mathcal{A}$ contains two letter cells, one delay cell, and three state cells to simulate the two possible inputs and three states of automaton $\mathcal{A}$. It has only two meaningful attractors corresponding to the two successful cycles of automaton $\mathcal{A}$.

non-self-dual networks such that $\mathcal{N}_1 \equiv_W \mathcal{N}_2^{\mathfrak{C}}$. However, while the BRNN hierarchy has only height $\omega + 1$, the complete BRNN hierarchy shows a height of ω^ω levels. In fact, the complete BRNN hierarchy consists of an infinite alternating succession of pairs of non-self-dual and single self-dual classes, with non-self-dual classes at each limit level, as illustrated in Figure 31.

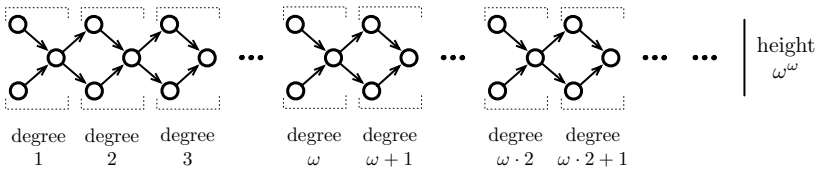


Figure 31 – The complete BRNN hierarchy: a transfinite alternating succession of pairs of non-self-dual classes of networks followed by single self-dual classes of networks, with non-self-dual classes at each limit level.

For convenience reasons, the degree $d(\mathcal{N})$ of a network $\mathcal{N}$ in the complete BRNN hierarchy is also defined such that the same degree is shared by both non-self-dual networks at some level and self-dual networks located just one level higher, namely:

$$d(\mathcal{N}) = \begin{cases} 1 & \text{if } L(\mathcal{N}) = \emptyset \text{ or } \emptyset^{\mathfrak{C}}, \\ \sup\{d(\mathcal{M}) + 1 : \mathcal{M} \text{ non-self-dual and } \mathcal{M} <_W \mathcal{N}\} & \text{if } \mathcal{N} \text{ is non-self-dual}, \\ \sup\{d(\mathcal{M}) : \mathcal{M} \text{ non-self-dual and } \mathcal{M} <_W \mathcal{N}\} & \text{if } \mathcal{N} \text{ is self-dual}. \end{cases}$$

The isomorphism between the Wagner hierarchy and the complete BRNN hierarchy ensures that the complete BRNN hierarchy is also decidable, i.e., that there exists an algorithmic procedure which computes the degree of any network. The following theorem summarizes the properties of the complete BRNN hierarchy.

Theorem 41. *The complete BRNN hierarchy is a decidable pre-well-ordering of width* 2 *and height* ω^ω.

Proof. The Wagner hierarchy consists of a decidable pre-well-ordering of width 2 and height ω^ω [182]. Proposition 40 ensures that the complete BRNN hierarchy and the Wagner hierarchy are isomorphic. □

A detailed description of the decidability procedure of the complete BRNN hierarchy can be given. In fact, the degree of a network $\mathcal{N}$ in the BRNN hierarchy corresponds precisely to the largest ordinal α such that there exists an alternating tree or a co-alternating tree of length α in the deterministic Muller automaton $\mathcal{A}_\mathcal{N}$.

Theorem 42. *Let $\mathcal{N}$ be some Boolean recurrent network provided with a type specification of its attractors, $\mathcal{A}_\mathcal{N}$ be the corresponding deterministic Muller automaton of $\mathcal{N}$, and α be an ordinal such that $0 < \alpha < \omega^\omega$.*

- *If there exists in $\mathcal{A}_\mathcal{N}$ a maximal alternating tree of length α and no maximal co-alternating tree of length α, then $d(\mathcal{N}) = \alpha$ and $\mathcal{N}$ is non-self-dual.*

- *If there exists in $\mathcal{A}_\mathcal{N}$ a maximal co-alternating tree of length α and no maximal alternating tree of length α, then $d(\mathcal{N}) = \alpha$ and $\mathcal{N}$ is non-self-dual.*
- *If there exist in $\mathcal{A}_\mathcal{N}$ both a maximal alternating tree as well as a maximal co-alternating tree of length α, then $d(\mathcal{N}) = \alpha$ and $\mathcal{N}$ is self-dual.*

Proof. By Proposition 40, the degree of a network $\mathcal{N}$ in the complete BRNN hierarchy is equal to the degree of its corresponding deterministic Muller automaton $\mathcal{A}_\mathcal{N}$ in the Wagner hierarchy. Moreover, this latter corresponds precisely to the length of a maximal alternating or co-alternating tree contained in this automaton [145, 182]. □

By Theorem 42, the decidability procedure of the degree of a neural network $\mathcal{N}$ in the complete BRNN hierarchy consists in firstly translating the network $\mathcal{N}$ into its corresponding deterministic Muller automaton $\mathcal{A}_\mathcal{N}$, as described in Proposition 35, and secondly computing the ordinal $\alpha < \omega^\omega$ corresponding to the length of a maximal alternating tree, or co-alternating tree, contained in $\mathcal{A}_\mathcal{N}$. Note that this procedure can be achieved by some graph analysis of the automaton $\mathcal{A}_\mathcal{N}$, since this latter is always finite.

Accordingly, the induced refined measure of complexity for Boolean neural networks, by being related to the notion of (co-)alternating trees, and hence of cycles, of the underlying Muller automata, is also in turn directly associated to the intricacy of the sets of attractors of the networks.

More precisely, the ω first levels of the complete BRNN hierarchy provide a classification of the networks that might alternate at most *finitely* many times between different types of attractors along their possible computations.[7] The levels of transfinite degrees correspond to some refined classification of the collection of networks that are able to alternate *infinitely* many times between different types of attractors.[8] The more intricate the structure of their attractors – in terms of inclusion and accessibility relation –, the more complex the networks.

The decidability procedure of the complete BRNN hierarchy is illustrated in the following Example 43.

Example 43. Let $\mathcal{N}$ be the network described in Figure 30(*a*). Then, by Theorem decidabilityM, $d(\mathcal{N}) = \omega^2$ and $\mathcal{N}$ is non-self-dual. Indeed, the following sequence of cycles

$$\left\{\begin{pmatrix}0\\0\\0\end{pmatrix}\right\} \subsetneq \left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}\right\} \subsetneq \left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}\right\}$$

is a maximal co-alternating tree of length ω^2 in the graph of the deterministic Muller automaton $\mathcal{A}_\mathcal{N}$ depicted in Figure 30(*b*).

Finally, it can be remarked that if some given Muller automaton contains either an alternating or a co-alternating tree of length α, then it also necessarily contain

[7] These are indeed the attractor dynamics associated to the (co)-alternating trees if lengths n, for $n \in \mathbb{N}^*$.

[8] Indeed, for any ordinal α such that $\omega \leq \alpha < \omega^\omega$, a maximal alternating or co-alternating tree of length α necessarily contains at least two cycles c_1 and c_2 such that $c_1 \subsetneq c_2$ and c_1 is successful iff c_2 is non-successful. Hence, there necessarily exists a path that can alternate infinitely many times between c_1 and c_2.

both an alternating and a co-alternating tree of length β, for all $\beta < \alpha$.[9] Therefore, any network of the complete BRNN hierarchy does contain in its dynamical structure all the possible attractor dynamics of every other networks of strictly smaller degrees. It would be able to mimic, in terms of attractor dynamics, the computations of every less complicated network. Besides, this concept of alternation between different types of attractors appears to be linked to the transient trajectories between attractor basins, a concept referred to as "itinerancy" in the literature [79, 84, 166, 167, 168].

Comparison Between the BRNN and the Complete BRNN hierarchies

We defined the BRNN hierarchy as a classification of Boolean networks whose attractors' types are induced by the existence of an output layer. The complete BRNN hierarchy consists of a similar classification of Boolean networks whose attractors' type specification has been totally relaxed. In fact, the BRNN hierarchy can be seen as a proper initial segment of the complete BRNN hierarchy: the networks of BRNN hierarchy and those of the initial segment of length $\omega + 1$ of the complete BRNN hierarchy recognize the same languages.

Proposition 44. *Let $L \subseteq (\mathbb{B}^k)^\omega$. Then L is recognizable by some network $\mathcal{N}$ of the BRNN hierarchy iff L is also recognizable by some network $\mathcal{N}'$ of the complete BRNN hierarchy such that either $d(\mathcal{N}') < \omega$ or $d(\mathcal{N}') = \omega$ and $\mathcal{N}'$ contains a maximal co-alternating tree of length ω but no maximal alternating tree of length ω.*

Proof. Given any deterministic Muller automaton $\mathcal{A}$, let the degree of $\mathcal{A}$ in the Wagner hierarchy be denoted by $d_W(\mathcal{A})$. Then, the relationship between the DBA and the Wagner hierarchies ensures that L is recognizable by some deterministic Büchi automaton iff L is also recognizable by some deterministic Muller automaton $\mathcal{A}$ such that either $d_W(\mathcal{A}) < \omega$ or $d_W(\mathcal{A}) = \omega$ and $\mathcal{A}$ contains a maximal co-alternating tree of length ω but no maximal alternating tree of length ω [134]. Theorems 28 and 37 together with Proposition 40 allow to transpose these results to the neural network context, and therefore lead to the conclusion. □

The discrepancy between the RNN the complete BRNN hierarchies is explained by the type specification of the attractors of the networks. Indeed, the networks whose attractors' types are determined by some output layer will never contain any maximal alternating or co-alternating chain of length strictly larger than ω in their underlying automata.[10] By contrast, the networks whose attractors' types are freely determined can contain maximal alternating or co-alternating trees of length up to ω^ω. Moreover, it can be observed that for any ordinal $\alpha \leq \omega$, the two notions of alternating chain and alternating tree of length α coincide. Therefore, according to Theorems 33 and 42, the decidability procedures of the the BRNN hierarchy and of the initial segment of length $\omega + 1$ of the complete BRNN hierarchy reduce to

[9] This observation follows from the definition of an alternating tree.

[10] Indeed, suppose that A_1 and A_2 are two attractors of $\mathcal{N}$ whose type specifications have been determined by some output layer. Suppose that A_1 is meaningful and that $A_1 \subseteq A_2$. It necessarily follows that A_2 is also meaningful. By transposing the reasoning on the cycles of the underlying automaton, it follows that this latter can never involve alternating or co-alternating trees of length strictly larger than ω (ω^1 for one alternation).

the very same. The comparison between the two hierarchies is illustrated in Figure 32.

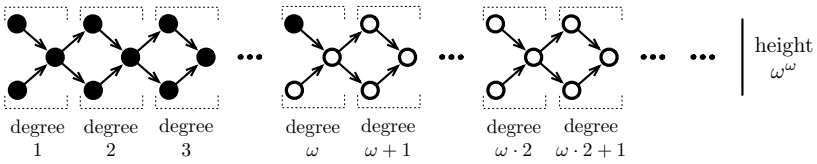


Figure 32 – Comparison between the RNN and the complete BRNN hierarchies. The BRNN hierarchy, depicted by the sequence of blacks classes, consists of an initial segment of length $\omega + 1$ of the complete BRNN hierarchy, which has height ω^ω.

7.3.4 Application

The Basal Ganglia-Thalamocortical Network

We provide an application of our attractor-based complexity measure to a case study. For this purpose, we consider one of the main systems of the brain which is involved in information processing: the basal ganglia-thalamocortical network. This network has been investigated for many years, in particular in relation to disorders of the motor system and of the sleep-waking cycle, see for instance [6, 7, 8, 9, 83, 93, 94, 121, 162].

More generally, we assume that the encoding of a large amount of the information treated by the brain is performed by recurrent patterns of activity circulating in the information transmitting system of this network. For this reason, we focus our attention on the complexity of the dynamics that may emerge from that system. We consider a Boolean recurrent neural network model of the information transmitting system of the basal ganglia-thalamocortical network, and study the attractor-based complexity of this network.

The basal ganglia-thalamocortical network is formed by several parallel and segregated circuits involving different areas of the cerebral cortex: striatum, pallidum, thalamus, subthalamic nucleus and midbrain. A characteristic of all the circuits of the basal ganglia-thalamocortical network is a combination of "open" and "closed" loops with ascending sensory afferences reaching the thalamus and the midbrain, and with descending motor efferences from the midbrain (the tectospinal tract) and the cortex (the corticospinal tract). The pattern of connectivity corresponds to the wealth of data reported in the literature [8, 9, 70, 158].

We assume that each brain area is formed by a neural network and that the network of brain areas corresponding to the basal ganglia-thalamocortical network can be modeled by a Boolean neural network formed by 9 nodes: superior colliculus (SC), Thalamus, thalamic reticular nucleus (NRT), Cerebral Cortex, the two functional parts (striatopallidal and the striatonigral components) of the striatum (Str), the subthalamic nucleus (STN), the external part of the pallidum (GPe), and the output nuclei of the basal ganglia formed by the GABAergic projection neurons of the intermediate part of the pallidum and of the substantia nigra pars reticulata

(GPi/SNR).

We consider the ascending sensory pathway (IN), that reaches SC and the Thalamus. SC sends a projection outside of the system (OUT), to the motor system. The thalamus sends excitatory connections within the system via the thalamo-pallidal, thalamo-striatal and thalamo-cortical projections. Notice that STN receives also an excitatory projection from the Thalamus. NRT receives excitatory collateral projections from both the thalamo-cortical and cortico-thalamic projections. In turn, NRT sends an inhibitory projection to the Thalamus. The Cerebral Cortex receives also an excitatory input from STN and sends corticofugal projections to the basal ganglia (striatum and STN), to the thalamus, to the midbrain and to the motor system (OUT). The only excitatory nucleus of the basal ganglia is STN, that sends projections to the Cerebral Cortex, to GPe and to GPi/SNR. In the striatum (Str) the striatopallidal neurons send inhibitory projections to GPe and the striatonigral neurons send inhibitory projections to GPi/SNR, via the so-called "direct" pathway. The pallidum (GPe) plays a paramount role because it is an inhibitory nucleus, with reciprocal connections back to the striatum and to STN and a downstream inhibitory projection to GPi/SNR via the so-called "indirect" pathway. It is interesting to notice the presence of inhibitory projections that inhibit the inhibitory nuclei within the basal ganglia, thus leading to a kind of "facilitation", but also inhibitory projections that inhibit RTN, that is a major nucleus in regulating the activity of the thalamus. Our Boolean model of the basal ganglia-thalamocortical network is illustrated in Figure 33 and its connectivity patterns given in Table 4.

For sake of simplicity, we assume that all the nodes of the network are Boolean and that the two inputs are merged into one Boolean input node. Hence, one obtains a Boolean model of the basal-ganglia thalamocortical network with 9 activation nodes and 1 input node. [11]

Table 4 – The adjancency matrix of the Boolean model of the basal ganglia-thalamocortical network.

Source		**Target**									
Node #	Name	IN	SC	Thal.	RTN	GPi/SNr	STN	GPe	Str-D2	Str-D1	CCortex
0	IN	·	1	1	·	·	·	·	·	·	·
1	SC	int_1	·	1	·	·	·	·	·	·	·
2	Thal.	·	·	·	1	·	1	1	1	1	1
3	RTN	·	·	-1	·	·	·	·	·	·	·
4	GPi/SNr	·	-1	-1	-1	·	·	·	·	·	·
5	STN	·	·	·	·	2	·	2	·	·	2
6	GPe	·	·	·	-1/2	-1/2	-1/2	·	-1/2	-1/2	·
7	Str-D2	·	·	·	·	·	·	-1	·	·	·
8	Str-D1	·	·	·	·	-1/2	·	-1/2	·	·	·
9	CCortex	int_2	1/2	1/2	1/2	·	1/2	·	1/2	1/2	·

[11] Each of the 9 activation nodes of the network can be either active or quiet. Hence, there are $2^9 = 512$ possible network states, each one being represented by a 9-dimensional Boolean vector describing the sequence of active and quiet activation nodes. For example, the network state $(0, 1, 0, 0, 1, 1, 1, 1)$ means that the nodes #1 (SC), #3 (RTN) and #4 (GPi/SNR) are quiet, whereas every other activation node is active.

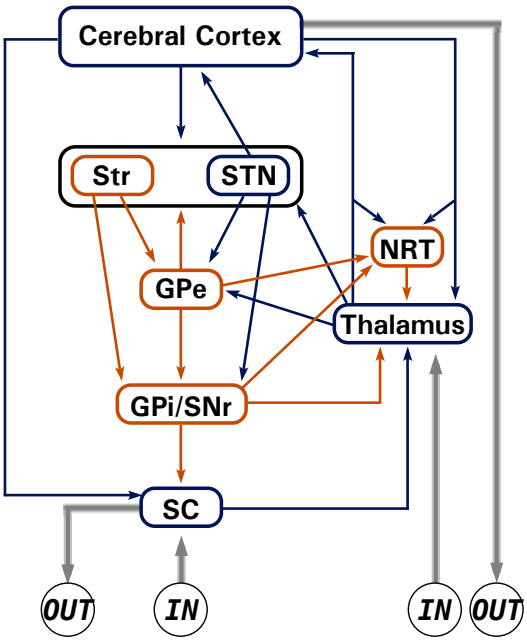


Figure 33 – Boolean model of the basal ganglia-thalamocortical network. The network is constituted of 9 different interconnected brain areas, each one represented by a single node in the Boolean neural network model: superior colliculus (SC), Thalamus, thalamic reticular nucleus (NRT), Cerebral Cortex, the striatopallidal and the striatonigral components of the striatum (Str), the subthalamic nucleus (STN), the external part of the pallidum (GPe), and the output nuclei of the basal ganglia formed by the GABAergic projection neurons of the intermediate part of the pallidum and of the substantia nigra pars reticulata (GPi/SNR). We consider also the inputs (IN) from the ascending sensory pathway and the motor outputs (OUT). The excitatory pathways are labeled in blue and the inhibitory ones in orange.

Attractor-Based Complexity

We now compute the attractor-based complexity measure of our Boolean model of the basal ganglia-thalamocortical network $\mathcal{N}$. Since the behaviour of network $\mathcal{N}$ is not determined by any designated output layer, the attractor-based complexity of $\mathcal{N}$ will be measured with respect to the complete BRNN hierarchy rather than with respect to the BRNN hierarchy. According to these considerations, as mentioned in Theorem 42, the attractor-based complexity of network $\mathcal{N}$ relies on the graphical structure of its corresponding deterministic Muller automaton $\mathcal{A}_{\mathcal{N}}$.

The deterministic Muller automaton $\mathcal{A}_{\mathcal{N}}$ associated to the network $\mathcal{N}$ contains $2^9 = 512$ states (numbered from 0 to 511) and $512 \times 2 = 1024$ transitions (512 labelled by 0 and 512 labelled by 1).[12] The automaton is illustrated in Figure 34(a). It contains only one strongly connected component $\mathcal{C}$, which corresponds to the subgraph induced by all states reachable from the initial state 0.[13] By Theorem 42, the attractor-based complexity of $\mathcal{N}$ is precisely determined by the cyclic struc-

[12] cf. Proposition 35 for the construction of $\mathcal{A}_{\mathcal{N}}$.

[13] We recall that a directed graph is called strongly connected if there is a path from every vertex of the graph to every other vertex.

ture (aternating-trees) of $\mathcal{C}$. The strictly connected component is illustrated in Figure 34(b).

This strongly connected component $\mathcal{C}$ corresponds to the subgraph of $\mathcal{A}_\mathcal{N}$ constituted by all states reachable from the initial state 0. In other words, any state of $\mathcal{A}_\mathcal{N}$ outside the strongly connected component $\mathcal{C}$ cannot be reached from the initial state 0, meaning that it can never occur in the dynamics of network $\mathcal{N}$ starting from initial state 0, and hence plays no role in the attractor-based complexity of network $\mathcal{N}$. In fact, the attractor-based complexity of network $\mathcal{N}$ will be precisely determined by the cyclic structure of the strongly connected component $\mathcal{C}$ of automaton $\mathcal{A}_\mathcal{N}$.

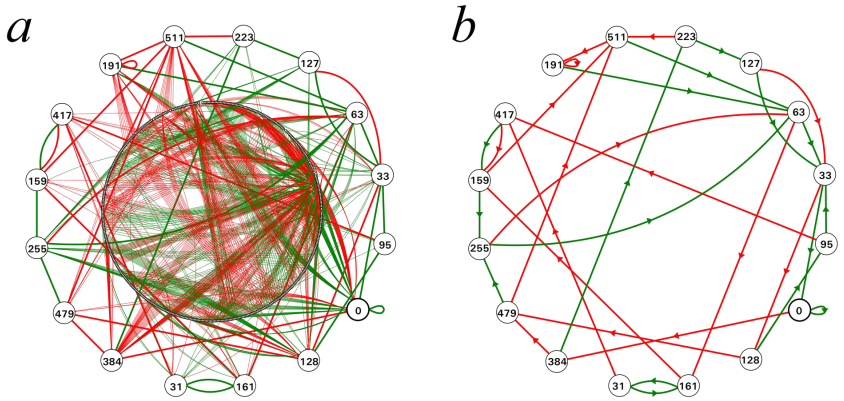


Figure 34 – Deterministic Muller automaton based on the "basal ganglia-thalamocortical" network of Figure 33. **a.** The graph of the automaton $\mathcal{A}_\mathcal{N}$ associated to network $\mathcal{N}$ contains 512 states and 1024 directed transitions. The colours of the transitions represent their labels: green for label 0 and red for label 1. For sake of readability, the directions of the transitions have been removed. The states and transitions of the strongly connected component $\mathcal{C}$ of $\mathcal{A}_\mathcal{N}$ have been pulled out of the central graph and drawn in larger font. **b.** The graph of the strongly connected component $\mathcal{C}$ of $\mathcal{A}_\mathcal{N}$.

In order to complete the description of the Muller automaton $\mathcal{A}_\mathcal{N}$, we further need to specify its table, i.e., to determine among all possible cycles which ones are successful and which ones are non-successful.[14] According to the biunivocal correspondence between cycles of $\mathcal{A}_\mathcal{N}$ and attractors of $\mathcal{N}$, this amounts to assigning among all possible attractors of $\mathcal{N}$ which ones are meaningful and which ones are spurious. For this purpose, we have computed the list of all cycles of the strictly connected component $\mathcal{C}$, and for each cycle, we have further computed its decomposition into constitutive cycles (cycles which do not visit the same vertex two times). The results are summarized in Table 5. Then, we have assigned a type specification to each cycle of $\mathcal{C}$ according to the following neurobiological criteria: First, a constitutive cycle is considered to be spurious if it is characterized either by active SC and quiet Thalamus at the same time step or by a quiet GPi/SNR during the majority of the duration of the constitutive cycle. A constitutive cycle is meaningful otherwise. Secondly, a non-constitutive cycle is considered to be meaningful

[14] Note that since every cycle of $\mathcal{A}_\mathcal{N}$ is by definition contained in a strongly connected component and since $\mathcal{C}$ is the only strongly connected component of $\mathcal{A}_\mathcal{N}$, all cycles of $\mathcal{A}_\mathcal{N}$ are necessarily contained in $\mathcal{C}$.

if it contains a majority of meaningful constitutive cycles, and spurious if it contains a majority of spurious constitutive cycles – and in the case of it containing as much meaningful as spurious constitutive cycles, its type specification was chosen to be meaningful. In order to illustrate this procedure, let us consider for example the cycles starting from state 0. Table 5 shows that there are overall 68 cycles and 24 constitutive cycles starting from state 0. An example of this assignment procedure is illustrated in Figure 35.

Table 5 – Number of cycles and constitutive cycles starting from each state of the strongly connected component $\mathcal{C}$.

State	# cycles	# constitutive cycles
0	68	24
31	47	20
33	87	24
63	93	21
95	39	21
127	21	17
128	63	24
159	77	22
161	72	20
191	52	19
223	43	21
255	53	17
384	67	24
417	35	20
479	48	16
511	84	21

After the type specification to every cycle has been assigned, the attractor-based complexity of the network $\mathcal{N}$ can be explicitly computed. Theorem 42 ensures that the degree of $\mathcal{N}$ is given by the length of a maximal (co-)alternating tree contained in $\mathcal{A}_\mathcal{N}$, or more precisely, contained in $\mathcal{C}$.[15] After an exhaustive analysis of the strongly connected component $\mathcal{C}$, we found no maximal alternating trees with a length above ω^5. However, we found 3 maximal co-alternating trees of length ω^6. One such maximal co-alternating tree is illustrated in Figure 36. According to these considerations, it follows from Theorem 42 that the attractor-based complexity of network $\mathcal{N}$ is ω^6 and that $\mathcal{N}$ is non-self-dual.

As already mentioned, the attractor-based complexity measure is linked to the intricacy of the set of attractors of the network, and more precisely, to the maximal number of times that a network might alternate between meaningful and spurious attractors along some computation. This feature refers to the ability of the networks to perform more or less complicated classification tasks of their input streams via

[15] Indeed, since $\mathcal{A}_\mathcal{N}$ contains only one strongly connected component $\mathcal{C}$, the maximal (co-)alternating tree of $\mathcal{A}_\mathcal{N}$ is necessarily contained in $\mathcal{C}$. The reason is that every cycle of $\mathcal{A}_\mathcal{N}$ is, by being a cycle, necessarily contained in a strongly connected component of $\mathcal{A}_\mathcal{N}$. And since $\mathcal{C}$ is the only such component, every cycle is contained in $\mathcal{C}$. Hence, the whole maximal (co-)alternating tree, by being composed of several cycles, is also necessarily contained in $\mathcal{C}$.

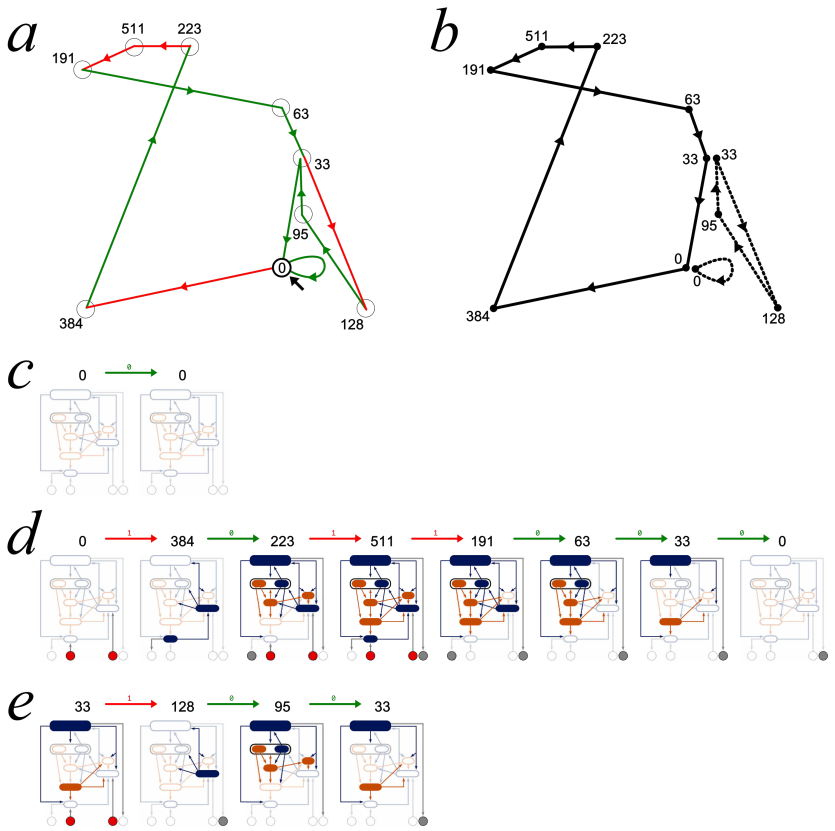


Figure 35 – A cycle and its constitutive cycles. **a.** A cycle starting from state 0 given by the states $(0, 0, 384, 223, 511, 191, 63, 33, 128, 95, 33, 0)$. **b.** This cycle contains three constitutive cycles $(0, 0)$, (0, 384, 223, 511, 191, 63, 33, 0) and (33, 128, 95, 33) that were assigned with type specification spurious (dotted line), meaningful (solid line), and spurious (dotted line), respectively. **c.** Sequence of states (with graphical representation of the corresponding activated nodes) of the basal ganglia-thalamocortical network corresponding to the spurious constitutive cycle $(0, 0)$. **d.** Sequence of states of the network corresponding to the meaningful constitutive cycle (0, 384, 223, 511, 191, 63, 33, 0). **e.** Sequence of states of the activated network corresponding to the spurious constitutive cycle (33, 128, 95, 33).

the manifestation of meaningful or spurious attractor dynamics. In this case, a degree of ω^6 signifies the existence of a sequence 7 attractors, one included into the other, that the network might alternatively visit along its computations. This disposition of attractors provides the possibility to discriminate (in a binary way) between input streams of an ω-regular language of degree ω^6 in the Wagner hierarchy. It can also reproduce the discrimination task associated to any ω-regular language of any strictly smaller degree.

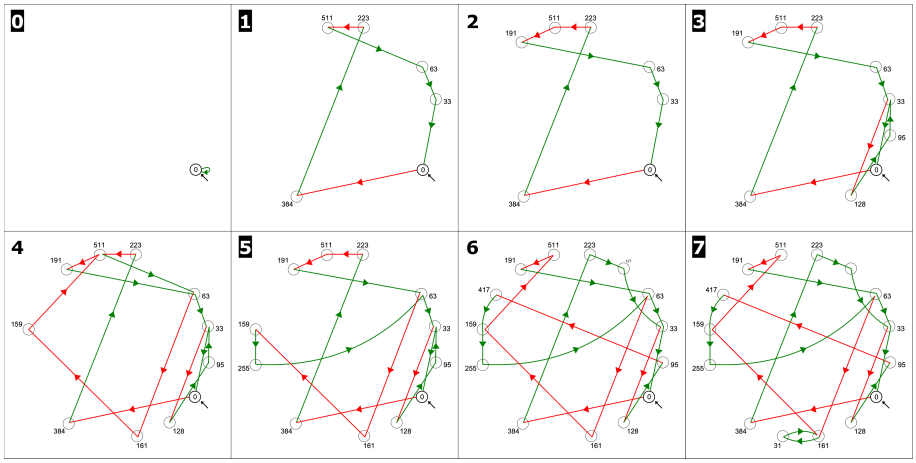


Figure 36 – A maximal co-alternating tree of the deterministic Muller automaton $\mathcal{A}_{\mathcal{N}}$. Panels 0 to 7 illustrate the sequence of eight cycles $(C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ one included into the next. Cycles C_0, C_1, C_3, C_5, and C_7 are spurious whereas cycles C_2, C_4, and C_6 are meaningful. The sequence of cycles $(C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ compose a maximal co-alternating tree of $\mathcal{A}_{\mathcal{N}}$. This maximal co-alternating tree contains 6 alternations between spurious and meaningful cycles, and thus has a length of ω^6. Therefore, the attractor-based degree of $\mathcal{N}$ equals ω^6.

7.4 Deterministic Sigmoidal Neural Networks

7.4.1 The Model

We now generalize the study to the case of sigmoidal neural networks. More precisely, we consider a general model of first-order recurrent neural networks, as described in Chapter 4, provided of three groups of neurons: a layer of Boolean "input" cells, which is connected to a set of recurrently interconnected sigmoidal "internal" neurons, itself connected to a layer of Boolean "output" cells. At each time step, the activation value of each neuron is given by an affine combination of the other cells' activation values and inputs.

The sigmoidal internal neurons introduce the biological source of nonlinearity which is so important to neural computation. They provide the possibility to surpass the capabilities of finite state automata, or even of Turing machines. The sigmoidal activation functions are particularly appropriate for the implementation of various learning algorithms. In neurobiology, they are usually considered as a representation of the rate of action potential firing in the cell. The Boolean input and output cells carry out the exchange of discrete information between the network and the environment. It will be noticed that, if some infinite input stream is supplied, the output cells necessarily enter into some attractor dynamics. The Boolean nature of the input and output cells provides the possibility to consider recurrent neural networks as computing systems working on discrete inputs and outputs streams, and consequently, to compare their computational capabilities to those of classical abstract machines, along the lines of [24, 26, 154, 155]. The expressive power of the networks will be related to the attractor dynamics of their

Boolean output cells.

Formally, a *sigmoidal deterministic (first-order) recurrent neural network* (simply denoted by D-RNN) contains M Boolean input cells $(u_i)_{i=1}^M$, N sigmoidal internal neurons $(x_i)_{i=1}^N$, and P Boolean output cells $(y_i)_{i=1}^P$. The dynamics of the network is computed as usual: given the activation values of the input and internal neurons $(u_j)_{j=1}^M$ and $(x_j)_{j=1}^N$ at time t, the activation values of each sigmoidal internal and Boolean output neuron x_i and y_i at time $t+1$ are updated by the following equations, respectively:

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t)\cdot u_j(t) + c_i(t)\right) \text{ for } i = 1,\ldots,N \quad (7.2)$$

$$y_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t)\cdot u_j(t) + c_i(t)\right) \text{ for } i = 1,\ldots,P \quad (7.3)$$

where $a_{ij}(t)$, $b_{ij}(t)$, and $c_i(t)$ are the weights of the synaptic connections and the bias of the network at time t, and σ and θ are the linear-sigmoid[16] and Heaviside step activation functions. Such a D-RNN is illustrated in Figure 37.

The dynamics of any D-RNN $\mathcal{N}$ is therefore given by the function $f_{\mathcal{N}} : \mathbb{B}^M \times \mathbb{B}^N \to \mathbb{B}^N \times \mathbb{B}^P$ defined by

$$f_{\mathcal{N}}(\boldsymbol{u}(t), \boldsymbol{x}(t)) = (\boldsymbol{x}(t+1), \boldsymbol{y}(t+1))$$

where the components of $\boldsymbol{x}(t+1)$ and $\boldsymbol{y}(t+1)$ are given by Equations (7.2) and (7.3), respectively.

Consider some D-RNN $\mathcal{N}$ provided with M Boolean input cells, N sigmoidal internal cells, and P Boolean output cells. For each time step $t \geq 0$, the *state* of $\mathcal{N}$ at time t consists if a pair of the form

$$\langle \boldsymbol{x}(t), \boldsymbol{y}(t) \rangle \in [0,1]^N \times \mathbb{B}^P.$$

The second element of this pair, namely $\boldsymbol{y}(t)$, is called the *output state* of $\mathcal{N}$ at time t.

Assuming the initial state of the network to be $\langle \boldsymbol{x}(0), \boldsymbol{y}(0) \rangle = \langle \boldsymbol{0}, \boldsymbol{0} \rangle$, any infinite input stream

$$s = (\boldsymbol{u}(t))_{t\in\mathbb{N}} = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots \in (\mathbb{B}^M)^\omega$$

induces via Equations (7.2) and (7.3) an infinite sequence of consecutive states

$$c_s = (\langle \boldsymbol{x}(t), \boldsymbol{y}(t) \rangle)_{t\in\mathbb{N}} = \langle \boldsymbol{x}(0), \boldsymbol{y}(0) \rangle \langle \boldsymbol{x}(1), \boldsymbol{y}(1) \rangle \cdots \in ([0,1]^N \times \mathbb{B}^P)^\omega$$

called the *computation* of $\mathcal{N}$ induced by s. The corresponding infinite sequence of output states

$$c'_s = (\boldsymbol{y}(t))_{t\in\mathbb{N}} = \boldsymbol{y}(0)\boldsymbol{y}(1)\boldsymbol{y}(2)\cdots \in (\mathbb{B}^P)^\omega$$

is the *Boolean computation* of $\mathcal{N}$ induced by s. The computation of such a D-RNN is illustrated in Figure 37.

[16] The linear-sigmoid activation function is a simple piecewise linear approximation of a sigmoidal activation function. However, the results of this section remain valid for any other kind of sigmoidal activation function satisfying the properties mentioned in [88, Section 4].

Note that any D-RNN $\mathcal{N}$ (with P Boolean output cells) can only have 2^P – i.e., finitely many – possible distinct output states. Hence, by a similar argument as in Section 7.3.1, for any infinite Boolean computation c'_s, there necessarily exists at least one output state that recurs infinitely often in c'_s. In fact, any Boolean computation c'_s necessarily consists of a finite prefix of output states followed by an infinite suffix of output states that repeat infinitely often – yet not necessarily in a periodic manner. The non-empty set of all the output states that repeat infinitely often in c'_s is denoted by $\inf(c'_s)$. Following Definition 24, a set of states of the form $\inf(c'_s) \subseteq \mathbb{B}^P$ will be called an *attractor* for $\mathcal{N}$. In words, an attractor of $\mathcal{N}$ is a set of output states into which the Boolean computation of the network could become forever trapped – yet not necessarily in a periodic manner. An attractor of some D-RNN is illustrated in Figure 37.

Once again, we suppose that the attractors are of two distinct types, either *meaningful* or *spurious*, and from this point onwards, we assume that any D-RNN is equipped with a corresponding classification of all of its attractors into meaningful and spurious types.

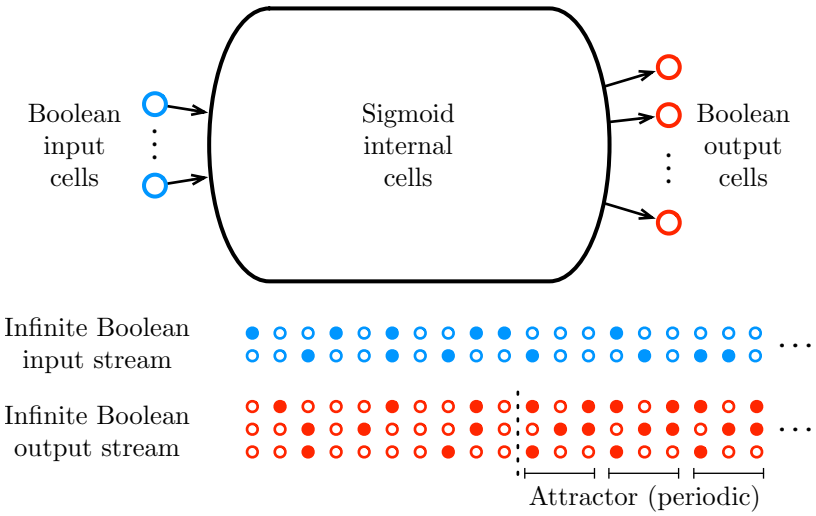


Figure 37 – Illustration of the computational process performed by some D-RNN. The infinite Boolean input stream $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots \in (\mathbb{B}^M)^\omega$, represented by the blue pattern, induces a corresponding Boolean output stream – or Boolean computation – $c'_s = \boldsymbol{y}(0)\boldsymbol{y}(1)\boldsymbol{y}(2)\cdots \in (\mathbb{B}^P)^\omega$, represented by the red pattern. The filled and empty circles represent active and quiet Boolean cells, respectively. From some time step onwards, a certain set of output states begins to repeat infinitely often: this corresponds the attractor associated to the input stream s. If the attractor occurs in a periodic manner, it corresponds to some spatiotemporal pattern.

Given some D-RNN $\mathcal{N}$, an infinite input stream $s \in (\mathbb{B}^M)^\omega$ of $\mathcal{N}$ is called *meaningful* if $\inf(c'_s)$ is a meaningful attractor, and it is called *spurious* if $\inf(c'_s)$ is a spurious attractor. The set of all meaningful input streams of $\mathcal{N}$ is called the *neural ω-language recognized by $\mathcal{N}$* and is denoted by $L(\mathcal{N})$. A set $L \subseteq (\mathbb{B}^M)^\omega$ is said to be *recognizable* by some D-RNN if there exists a network $\mathcal{N}$ such that $L(\mathcal{N}) = L$.

Now, six different models of D-RNNs can be considered according to the nature of their synaptic weights:

1. the *static rational D-RNNs* (D-St-RNN[$\mathbb{Q}$]s) refer to the class of all D-RNNs whose every weights are static and modelled by rational values.
2. the *static real (or analog) D-RNNs* (D-St-RNN[$\mathbb{R}$]s) refer to the class of all D-RNNs whose every weights are static and modelled by real values.
3. the *bi-valued evolving rational D-RNNs* (D-Ev$_2$-RNN[$\mathbb{Q}$]s) refer to the class of all D-RNNs whose every evolving weights are bi-valued and every static weights are rational.
4. the *bi-valued evolving real D-RNNs* (D-Ev$_2$-RNN[$\mathbb{R}$]s) refer to the class of all D-RNNs whose every evolving weights are bi-valued and every static weights are real.
5. the *(general) evolving rational D-RNNs* (D-Ev-RNN[$\mathbb{Q}$]s) refer to the class of all D-RNNs whose every evolving and static weights are rational.
6. the *(general) evolving real D-RNNs* (D-Ev-RNN[$\mathbb{R}$]s) refer to the class of all D-RNNs whose every evolving and static weights are real.

The following strict inclusions, illustrated in Figure 38, hold by definition:

$$\begin{array}{ccccc} \text{D-St-RNN}[\mathbb{Q}]\text{s} & \subsetneq & \text{D-Ev}_2\text{-RNN}[\mathbb{Q}]\text{s} & \subsetneq & \text{D-Ev-RNN}[\mathbb{Q}]\text{s} \\ \not\cap & & \not\cap & & \not\cap \\ \text{D-St-RNN}[\mathbb{R}]\text{s} & \subsetneq & \text{D-Ev}_2\text{-RNN}[\mathbb{R}]\text{s} & \subsetneq & \text{D-Ev-RNN}[\mathbb{R}]\text{s} \end{array}$$

7.4.2 Expressive Power

We now provide a complete characterization of the expressive powers of deterministic sigmoidal recurrent neural networks [30]. We show that D-St-RNN[$\mathbb{Q}$]s are computationally equivalent to Muller Turing machines, and hence, possess an expressive power strictly inside the class of $BC(\mathbf{\Pi}_2^0)$ neural ω-languages (Theorem 45). The five other models of D-Ev$_2$-RNN[$\mathbb{Q}$]s, D-Ev-RNN[$\mathbb{Q}$]s, D-St-RNN[$\mathbb{R}$]s, D-Ev$_2$-RNN[$\mathbb{R}$]s, D-Ev-RNN[$\mathbb{R}$]s are equivalent to each other and strictly more powerful than deterministic Muller Turing machines, with an expressive power equal to the class of $BC(\mathbf{\Pi}_2^0)$ neural ω-languages (Theorem 46). In this sense, these five neural models of computation are *super-Turing*. These results are summarized in Table 6 and Figure 38.

We first characterize the computational power of D-St-RNN[$\mathbb{Q}$]s.

Theorem 45. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language. Then L is recognizable by some D-St-RNN[$\mathbb{Q}$] if and only if L is recognizable by some deterministic Muller TM. In particular, if L is recognizable by some D-St-RNN[$\mathbb{Q}$], then $L \in BC(\mathbf{\Pi}_2^0)$.*

Proof. Let $\mathcal{N}$ be some D-St-RNN[$\mathbb{Q}$] recognizing the neural ω-language $L(\mathcal{N})$. Since the synaptic weights of $\mathcal{N}$ are rational and remain constant over time, Equations (7.2) and (7.3) are recursive, and hence, the function $f_{\mathcal{N}}$ defined by $f_{\mathcal{N}}(\boldsymbol{u}(t), \boldsymbol{x}(t)) =$

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

Table 6 – Expressive power of the six models of D-RNNs.

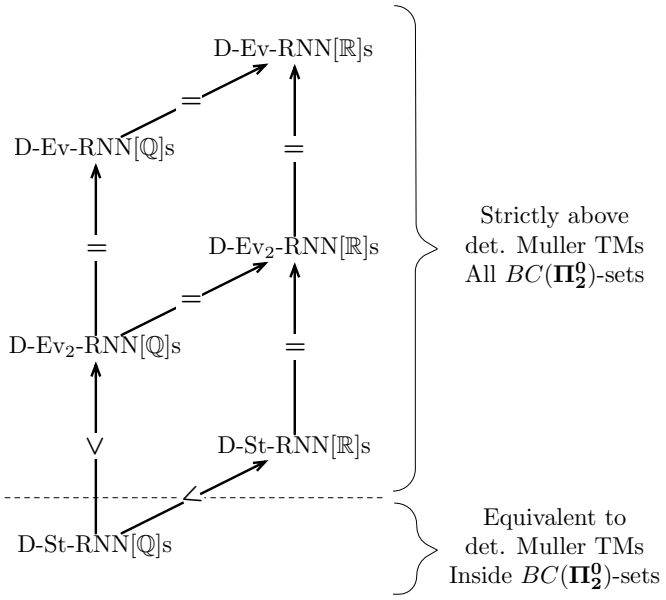


Figure 38 – Relationships between the expressive powers of the six models of first-order D-RNNs. The directed arrow correspond to the strict inclusions between the different classes of D-RNNs: there is an arrow from one model to the other if the former is less powerful than or equally powerful to the latter. The relation represented by these arrows is clearly transitive. In the sequel, we show that D-St-RNN[$\mathbb{Q}$]s are computationally equivalent to deterministic Muller Turing machines (Theorem 45), and that the five other models of D-Ev$_2$-RNN[$\mathbb{Q}$]s, D-Ev-RNN[$\mathbb{Q}$]s, D-St-RNN[$\mathbb{R}$]s, D-Ev$_2$-RNN[$\mathbb{R}$]s, D-Ev-RNN[$\mathbb{R}$]s are all equivalent to each other, and strictly more powerful than deterministic Muller Turing machines (Theorem 46). To illustrate these results, the arrows are labeled by a "$<$" or an "$=$" to designate if there is a strict inequality or an equivalence between the expressive powers of the respective models. The border between the Turing and super-Turing level is represented by the dashed line.

$(\boldsymbol{x}(t+1), \boldsymbol{y}(t+1))$ is also clearly recursive. Consequently, there exists some TM $\mathcal{M}$ with $N+P$ work tapes which can simulate the behavior of $\mathcal{N}$ by writing on its tapes the successive rational and Boolean activations values of the N and P internal and output cells of $\mathcal{N}$, respectively. We next provide $\mathcal{M}$ with 2^P additional desig-

nated states $q_1, \ldots, q_{2^P}$, and we modify its program in such a way that, after each simulation step, $\mathcal{M}$ enters state q_i iff $\mathcal{N}$ is in the i-th output state $\boldsymbol{b}_i \in \mathbb{B}^P$, according to the lexicographic order. In this way, each infinite input stream $s \in (\mathbb{B}^M)^\omega$ induces on the one side, in the network $\mathcal{N}$, a Boolean computation c'_s with an associated attractor $\inf(c'_s) \subseteq \mathbb{B}^P$, and, on the other side, in the machine $\mathcal{M}$, an infinite run ρ_s with an associated set of sates that are visited infinitely often $\inf(\rho_s)$ of the form $\inf(\rho_s) = Q \cup Q'$, with $Q' \subseteq \{q_1, \ldots, q_{2^P}\}$ and $Q' \neq \emptyset$. By construction, for any infinite input streams $s, s' \in (\mathbb{B}^M)^\omega$, the relation $\inf(c'_s) \neq \inf(c'_{s'}) \Rightarrow \inf(\rho_s) \neq \inf(\rho_{s'})$ holds. We can thus define the following Muller table of $\mathcal{M}$, namely $\mathcal{T} = \{\inf(\rho_s) \ : \ \inf(c'_s)$ is a meaningful attractor for $\mathcal{N}$, for any $s \in (\mathbb{B}^M)^\omega\}$.[17] According to this construction, one has $s \in L(\mathcal{N})$ iff $\inf(c'_s)$ is a meaningful attractor iff $\inf(\rho_s) \in \mathcal{T}$ iff $s \in L(\mathcal{M})$. Therefore, $L(\mathcal{N}) = L(\mathcal{M})$, showing that $L(\mathcal{N})$ is recognized by the deterministic Muller TM $\mathcal{M}$.

Conversely, let $\mathcal{M}$ be some deterministic Muller TM with table $\mathcal{T} = \{T_1, \ldots, T_k\}$ and associated ω-language $L(\mathcal{M})$. By the construction given in [155], there exists some (static) rational-weighted RNN $\mathcal{N}$ which can simulate the behavior of $\mathcal{M}$. Now, if $\mathcal{M}$ contains n states $q_1, \ldots, q_n$, we provide $\mathcal{N}$ with P additional Boolean output cells $y_1, \ldots, y_P$, with P satisfying $2^P \geq n$, and we update the simulation process such that, during the processing of the input stream, the machine $\mathcal{M}$ visits the state q_k iff the network $\mathcal{N}$ activates the k-th output state $\boldsymbol{b}_k$, according to the lexicographic order, for $k = 1, \ldots, n$. Next, for each element $T_i = \{q_{i_1}, \ldots, q_{i_{k(i)}}\}$ of the Muller table $\mathcal{T}$ of $\mathcal{M}$, we set the meaningful attractor $A_i = \{\boldsymbol{b}_{i_1}, \ldots, \boldsymbol{b}_{i_{k(i)}}\}$ in the network $\mathcal{N}$. All other possible attractors of $\mathcal{N}$ are considered to be spurious. In this way, for any infinite input stream $s \in (\mathbb{B}^M)^\omega$, the infinite run ρ_s of $\mathcal{M}$ satisfies $\inf(\rho_s) \in \mathcal{T}$ iff the Boolean computation c'_s of $\mathcal{N}$ satisfies that $\inf(c'_s)$ is a meaningful attractor. In other words, $s \in L(\mathcal{M})$ iff $s \in L(\mathcal{N})$. Therefore, $L(\mathcal{M}) = L(\mathcal{N})$, showing that $L(\mathcal{M})$ is recognized by the D-St-RNN[$\mathbb{Q}$] $\mathcal{N}$.

The second part of the Theorem comes from the previous equivalence and the fact that any ω-language recognized by some deterministic Muller TM is in $BC(\mathbf{\Pi_2^0})$ [134]. □

We now characterize the computational powers of the five remaining models of D-RNNs.

Theorem 46. *The five models of D-Ev$_2$-RNN[$\mathbb{Q}$]s, D-Ev-RNN[$\mathbb{Q}$]s, D-St-RNN[$\mathbb{R}$]s, D-Ev$_2$-RNN[$\mathbb{R}$]s, and D-Ev-RNN[$\mathbb{R}$]s are all super-Turing equivalent to each other, and recognize the class of $BC(\mathbf{\Pi_2^0})$ neural ω-languages. In other words, for any ω-language $L \subseteq (\mathbb{B}^M)^\omega$, the following conditions are equivalent:*

1. *$L \in BC(\mathbf{\Pi_2^0})$*
2. *L is recognizable by some D-Ev$_2$-RNN[$\mathbb{Q}$]*
3. *L is recognizable by some D-Ev-RNN[$\mathbb{Q}$]*
4. *L is recognizable by some D-St-RNN[$\mathbb{R}$]*

[17] Note that the relation $\inf(c'_s) \neq \inf(c'_{s'}) \Rightarrow \inf(\rho_s) \neq \inf(\rho_{s'})$ ensures that the table $\mathcal{T}$ is well defined, since it is impossible to have a situation where $\inf(c'_s)$ is meaningful, $\inf(c'_{s'})$ is spurious, and $\inf(\rho_s) = \inf(\rho_{s'})$, which would mean that $\inf(\rho_s) \in \mathcal{T}$, $\inf(\rho_{s'}) \notin \mathcal{T}$.

5. *L is recognizable by some D-Ev$_2$-RNN[$\mathbb{R}$]*

6. *L is recognizable by some D-Ev-RNN[$\mathbb{R}$]*

Proof. The proof is achieved via the two forthcoming Propositions 47 and 48.

First, let $L \subseteq (\mathbb{B}^M)^\omega$ such that $L \in BC(\mathbf{\Pi}_2^0)$. By Proposition 47, L is recognized by some D-Ev$_2$-RNN[$\mathbb{Q}$] and by some D-St-RNN[$\mathbb{R}$]. By definition, L is also recognizable by some D-Ev-RNN[$\mathbb{Q}$], D-Ev$_2$-RNN[$\mathbb{R}$], and D-Ev-RNN[$\mathbb{R}$] (cf. arrows in Figure 38). This proves the five implications from Point (1) to Points (2), (3), (4), (5), and (6).

Conversely, assume that L is recognizable by some D-Ev$_2$-RNN[$\mathbb{Q}$], some D-Ev-RNN[$\mathbb{Q}$], some D-St-RNN[$\mathbb{R}$], or some D-Ev$_2$-RNN[$\mathbb{R}$]. By definition, L is also recognizable by some D-Ev-RNN[$\mathbb{R}$] (cf. arrows in Figure 38). By Proposition 48, $L \in BC(\mathbf{\Pi}_2^0)$. This proves the five other implications from Points (2), (3), (4), (5), and (6) to Point (1). □

We now proceed to the proofs of Propositions 47 and 48.

Proposition 47. *Let $L \subseteq (\mathbb{B}^M)^\omega$. If $L \in BC(\mathbf{\Pi}_2^0)$, then L is recognizable by some D-St-RNN[$\mathbb{R}$] and by some D-Ev$_2$-RNN[$\mathbb{Q}$].*

Proof. First of all, let $L \subseteq (\mathbb{B}^M)^\omega$ such that $L \in \mathbf{\Pi}_2^0$. We will consider the case of $L \in BC(\mathbf{\Pi}_2^0)$ afterwards. Then L can be written as

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

where each $p_{i,j} \in (\mathbb{B}^M)^*$. Hence, a given infinite input $s \in (\mathbb{B}^M)^\omega$ belongs to L iff for all index $i \geq 0$ there exists an index $j \geq 0$ such that $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$, or equivalently, iff for all $i \geq 0$ there exists $j \geq 0$ such that $p_{i,j} \subsetneq s$. Besides, as described in details in [34], one can show that the infinite sequence $(p_{i,j})_{i,j \in \mathbb{N}}$ can be encoded into some single real number r_L such that, for any pair of indices $(i,j) \in \mathbb{N} \times \mathbb{N}$, the decoding procedure of $(r_L, i, j) \mapsto p_{i,j}$ is actually recursive.

According to these considerations, the problem of determining whether some input $s \in (\mathbb{B}^M)^\omega$ supplied step by step belongs to L or not can be decided in infinite time by the Algorithm 5 given below. This algorithm consists of two subroutines performed in parallel. It uses the designated real number r_L (on line 12), and it is designed in such a precise way that, on every input $s \in (\mathbb{B}^M)^\omega$, it returns infinitely many 1's iff $s \in \bigcap_{i \geq 0} \bigcup_{j \geq 0} p_{i,j} \cdot (\mathbb{B}^M)^\omega = L$. Moreover, note that if the designated real number r_L is provided in advance, then every step of Algorithm 5 is actually recursive.

Consequently, according to the real time computational equivalence between rational-weighted RNNs and TMs [155], there exists some RNN[$\mathbb{Q}$] $\mathcal{N}_1$ such that, if the real number r_L is given in advance as the activation value of one of its neuron x, then $\mathcal{N}_1$ is actually capable of simulating the behavior of Algorithm 5. In particular, to perform line 4, one uses M distinct cells in order to store the M Boolean components of $\boldsymbol{u}(t)$ (see [155] for further details). Consequently, if one adds to x a background synaptic connection of real intensity r_L, one obtains a RNN[$\mathbb{R}$] $\mathcal{N}_2$ capable of simulating Algorithm 5. Hence, if one further adds to $\mathcal{N}_2$ an additional

Boolean output cell y which is designed to take value 1 iff Algorithm 5 returns a 1, one obtains a D-St-RNN[$\mathbb{R}$] $\mathcal{N}$ such that, on every input $s \in (\mathbb{B}^M)^\omega$, the Boolean output cell y will produce infinitely many 1's iff Algorithm 5 will return infinitely many 1's, namely iff $s \in \bigcap_{i\geq 0} \bigcup_{j\geq 0} p_{i,j} \cdot (\mathbb{B}^M)^\omega = L$. Consequently, by taking $\{(1)\}$ and $\{(0),(1)\}$ as the two meaningful attractors of $\mathcal{N}$, one has $L(\mathcal{N}) = L$, meaning that L is recognized by some D-St-RNN[$\mathbb{R}$].

We now modify the proof in order to capture the case of a D-Ev$_2$-RNN[$\mathbb{Q}$]. In this context, one can show that the infinite sequence $(p_{i,j})_{i,j\in\mathbb{N}}$ can be encoded into some infinite word $w_L \in \{0,1\}^\omega$ such that, for any pair of indices $(i,j) \in \mathbb{N}\times\mathbb{N}$, the decoding procedure of $(w_L, i, j) \mapsto p_{i,j}$ is actually recursive. According to these considerations, we modify Algorithm 5 by assuming that it receives the designated infinite word w_L bit by bit instead of having the designated real number r_L be provided in advance. One then replaces lines 11 and 12 by the following two ones:

11: wait until $p_{i,j}$ has been encoded in w_L and until $c \geq |p_{i,j}|$
12: decode $p_{i,j}$ from w_L

Algorithm 5 can be performed by some D-Ev$_2$-RNN[$\mathbb{Q}$]. Indeed, in the D-St-RNN[$\mathbb{R}$] $\mathcal{N}$ described above, one replaces the static background activity of neuron x of real intensity r_L by an evolving background activity of intensities $w_L = w_L(0)w_L(1)w_L(2)\cdots \in \{0,1\}^\omega$. In this way, one obtains a network $\mathcal{N}'_1$ whose all

Algorithm 5 Procedure which uses the designated real number r_L

Require: Input $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots \in (\mathbb{B}^M)^\omega$ supplied step by step at successive time steps $t = 0, 1, 2, \ldots$

1: **SUBROUTINE 1**
2: $c \leftarrow 0$ // c counts the number of letters of s
3: **for all** time step $t \geq 0$ **do**
4: store each incoming Boolean vector $\boldsymbol{u}(t) \in \mathbb{B}^M$
5: $c \leftarrow c + 1$
6: **end for**
7: **END SUBROUTINE 1**

8: **SUBROUTINE 2**
9: $i \leftarrow 0$, $j \leftarrow 0$
10: **loop**
11: wait until $c \geq |p_{i,j}|$ // wait until $length(s) \geq length(p_{i,j})$
12: decode $p_{i,j}$ from r_L // recursive procedure if r_L is given in advance
13: **if** $p_{i,j} \subseteq s[0{:}c]$ **then** // $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$
14: **return** 1 // $\exists\ j$ s.t. $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$
15: $i \leftarrow i + 1$, $j \leftarrow 0$ // test if $s \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega$
16: **else** // $s \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega$
17: **return** 0 // $\neg\exists\ j' \leq j$ s.t. $s \in p_{i,j} \cdot (\mathbb{B}^M)^\omega$
18: $i \leftarrow i$, $j \leftarrow j + 1$ // test if $s \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega$
19: **end if**
20: **end loop**
21: **END SUBROUTINE 2**

static synaptic weights are rational and whose only evolving synaptic weight is bi-valued. We next slightly modify this networks in order to perform correctly the recursive updated lines 11 and 12 of Algorithm 5. One obtains a D-Ev$_2$-RNN[$\mathbb{Q}$] $\mathcal{N}'$ such that, on every input $s \in (\mathbb{B}^M)^\omega$, the Boolean output cell y of $\mathcal{N}'$ will produce infinitely many 1's iff the updated Algorithm 5 will return infinitely many 1's, namely iff $s \in \bigcap_{i\geq 0} \bigcup_{j\geq 0} p_{i,j} \cdot (\mathbb{B}^M)^\omega = L$. Therefore, by taking $\{(1)\}$ and $\{(0),(1)\}$ as the two sole meaningful attractors of $\mathcal{N}'$, one obtains $L(\mathcal{N}') = L$, meaning that L is recognized by some D-Ev$_2$-RNN[$\mathbb{Q}$].

This concludes the proof for the case of $L \in \mathbf{\Pi_2^0}$. We finally extend the proof for the case of $L \in BC(\mathbf{\Pi_2^0})$. We thus show that any finite union and any complementation of a $\mathbf{\Pi_2^0}$-set can also be recognized by some D-St-RNN[$\mathbb{R}$] and by some D-Ev$_2$-RNN[$\mathbb{Q}$].

First, let $L = L_1 \cup L_2$ such that $L_i \in \mathbf{\Pi_2^0}$, for $i = 1, 2$. By the previous arguments, there exist two D-St-RNN[$\mathbb{R}$]s (or two D-Ev$_2$-RNN[$\mathbb{Q}$]s) $\mathcal{N}_1$ and $\mathcal{N}_2$ which recognize L_1 and L_2, respectively. By suitably merging $\mathcal{N}_1$ and $\mathcal{N}_2$ into some new network $\mathcal{N}$, and by setting as meaningful attractors of $\mathcal{N}$ all those involving at least one output state for which at least one of the two Boolean output cells is spiking, one obtains a D-St-RNN[$\mathbb{R}$] (or a D-Ev$_2$-RNN[$\mathbb{Q}$]) $\mathcal{N}$ that recognizes $L_1 \cup L_2$. In other words, one has $L(\mathcal{N}) = L_1 \cup L_2 = L$.

Secondly, let $L \in \mathbf{\Sigma_2^0}$. Then by definition, $L^\complement \in \mathbf{\Pi_2^0}$. By the previous arguments, there exists a D-St-RNN[$\mathbb{R}$] (or a D-Ev$_2$-RNN[$\mathbb{Q}$]) $\mathcal{N}$ which recognizes $L^\complement$ via some relevant simulation of Algorithm 5. We now slightly update $\mathcal{N}$ in a way that its only Boolean output cell y takes value 1 iff Algorithm 5 returns a 0 (instead of 1). One thus obtains a D-St-RNN[$\mathbb{R}$] (or a D-Ev$_2$-RNN[$\mathbb{Q}$]) $\mathcal{N}'$ such that, on every input $s \in (\mathbb{B}^M)^\omega$, the Boolean output cell y of $\mathcal{N}'$ will produce infinitely many 1's as well as only finitely many 0's iff Algorithm 5 will return infinitely many 0's and finitely many 1's, i.e. iff $s \notin L^\complement$, i.e. iff $s \in L$. Consequently, by taking $\{(1)\}$ as the sole meaningful attractor of $\mathcal{N}'$, one obtains a D-St-RNN[$\mathbb{R}$] (or a D-Ev$_2$-RNN[$\mathbb{Q}$]) $\mathcal{N}'$ that recognizes L. In other terms, $L(\mathcal{N}') = L$.

According to these considerations, if $L \in BC(\mathbf{\Pi_2^0})$, then L is recognizable by some D-St-RNN[$\mathbb{R}$] and by some D-Ev$_2$-RNN[$\mathbb{Q}$]. □

Proposition 48. *Let $L \subseteq (\mathbb{B}^M)^\omega$. If L is recognizable by some D-Ev-RNN[$\mathbb{R}$], then $L \in BC(\mathbf{\Pi_2^0})$.*

Proof. Let $L \subseteq (\mathbb{B}^M)^\omega$ be recognizable by some D-Ev-RNN[$\mathbb{R}$] $\mathcal{N}$. Suppose that $\mathcal{N}$ contains the K meaningful attractors $A_i = \{\boldsymbol{b}_{i_1}, \ldots, \boldsymbol{b}_{i_{k(i)}}\}$, for $i = 1, \ldots, K$, where $1 \leq i_1 < \ldots < i_{k(i)} \leq 2^P$, and where $\boldsymbol{b}_n$ denotes the n-th Boolean vector of $\mathbb{B}^P$ according to the lexicographic order.

Note that the dynamics of $\mathcal{N}$ can naturally be associated with the function $g_\mathcal{N} : (\mathbb{B}^M)^\omega \to (\mathbb{B}^P)^\omega$ defined by $g_\mathcal{N}(s) = c'_s$, where $c'_s = \boldsymbol{y}(0)\boldsymbol{y}(1)\boldsymbol{y}(2)\cdots$ is the Boolean computation generated by $\mathcal{N}$ when the infinite input stream $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots$ is received. The nature of our dynamics ensures that this function is sequential, i.e., for any time step $t \geq 0$, the Boolean vectors $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t)$ are generated simultaneously. Hence, $g_\mathcal{N}$ is Lipschitz and thus continuous, cf. [34, Lemma 1].

Consequently, the neural ω-language $L(\mathcal{N})$ can be expressed as follows:

$$
\begin{aligned}
L(\mathcal{N}) &= \Big\{ s \in (\mathbb{B}^M)^\omega : \inf(c'_s) \text{ is a meaningful attractor} \Big\} \\
&= \Big\{ s \in (\mathbb{B}^M)^\omega : \inf(c'_s) = A_i \text{ for some } i = 1, \ldots, K \Big\} \\
&= \bigcup_{i=1}^{K} \Big\{ s \in (\mathbb{B}^M)^\omega : \inf(c'_s) = A_i \Big\} \\
&= \bigcup_{i=1}^{K} \Big\{ s \in (\mathbb{B}^M)^\omega : \text{for all } j \in \{i_1, \ldots, i_{k(i)}\}, \\
&\qquad\qquad g_{\mathcal{N}}(s) \text{ contains infinitely many } \boldsymbol{b}_j\text{'s, and} \\
&\qquad\qquad \text{for all } j \in \{1, \ldots, 2^P\} \setminus \{i_1, \ldots, i_{k(i)}\}, \\
&\qquad\qquad g_{\mathcal{N}}(s) \text{ contains finitely many } \boldsymbol{b}_j\text{'s} \Big\} \\
&= \bigcup_{i=1}^{K} \Big[\bigcap_{j \in \{i_1, \ldots, i_{k(i)}\}} \Big\{ s \in (\mathbb{B}^M)^\omega : g_{\mathcal{N}}(s) \text{ has infinitely many } \boldsymbol{b}_j\text{'s} \Big\} \cap \\
&\qquad \bigcap_{j \in \{1, \ldots, 2^P\} \setminus \{i_1, \ldots, i_{k(i)}\}} \Big\{ s \in (\mathbb{B}^M)^\omega : g_{\mathcal{N}}(s) \text{ has finitely many } \boldsymbol{b}_j\text{'s} \Big\} \Big] \\
&= \bigcup_{i=1}^{K} \Big[\bigcap_{j \in \{i_1, \ldots, i_{k(i)}\}} \Big\{ s \in (\mathbb{B}^M)^\omega : g_{\mathcal{N}}(s) \in \underbrace{\bigcap_{n \geq 0} \bigcup_{m \geq 0} (\mathbb{B}^P)^{n+m} \cdot \boldsymbol{b}_j \cdot (\mathbb{B}^P)^\omega}_{\substack{c'_s \text{ contains infinitely many } \boldsymbol{b}_j\text{'s, i.e.} \\ \forall n \geq 0\ \exists m \geq n\ c'_s(n+m) = \boldsymbol{b}_j \\ \text{thus in } \boldsymbol{\Pi}^0_2}} \Big\} \cap \\
&\qquad \bigcap_{j \in \{1, \ldots, 2^P\} \setminus \{i_1, \ldots, i_{k(i)}\}} \Big\{ s \in (\mathbb{B}^M)^\omega : g_{\mathcal{N}}(s) \in \underbrace{\Big(\bigcap_{n \geq 0} \bigcup_{m \geq 0} (\mathbb{B}^P)^{n+m} \cdot \boldsymbol{b}_j \cdot (\mathbb{B}^P)^\omega \Big)^{\complement}}_{\substack{c'_s \text{ contains only finitely many } \boldsymbol{b}_j\text{'s} \\ \text{complement of a } \boldsymbol{\Pi}^0_2\text{-set, thus in } \boldsymbol{\Sigma}^0_2}} \Big\} \Big] \\
&= \bigcup_{i=1}^{K} \Big[\bigcap_{j \in \{i_1, \ldots, i_{k(i)}\}} \underbrace{g_{\mathcal{N}}^{-1}\Big(\bigcap_{n \geq 0} \bigcup_{m \geq 0} (\mathbb{B}^P)^{n+m} \cdot \boldsymbol{b}_j \cdot (\mathbb{B}^P)^\omega \Big)}_{\text{preimage by a continuous function of a } \boldsymbol{\Pi}^0_2\text{-set, thus in } \boldsymbol{\Pi}^0_2} \cap \\
&\qquad \bigcap_{j \in \{1, \ldots, 2^P\} \setminus \{i_1, \ldots, i_{k(i)}\}} \underbrace{g_{\mathcal{N}}^{-1}\Big(\Big(\bigcap_{n \geq 0} \bigcup_{m \geq 0} (\mathbb{B}^P)^{n+m} \cdot \boldsymbol{b}_j \cdot (\mathbb{B}^P)^\omega \Big)^{\complement} \Big)}_{\text{preimage by a continuous function of a } \boldsymbol{\Sigma}^0_2\text{-set, thus in } \boldsymbol{\Sigma}^0_2} \Big]
\end{aligned}
$$

It follows that $L(\mathcal{N}) \in BC(\boldsymbol{\Pi}^0_2)$, since it consists of finite unions and finite intersections of $\boldsymbol{\Pi}^0_2$ and $\boldsymbol{\Sigma}^0_2$ sets. □

7.5 NONDETERMINISTIC SIGMOIDAL NEURAL NETWORKS

In this section, we extend the previous considerations to the nondeterministic context.

7.5.1 The Model: Neural Networks of Type 1

First, we consider the notion of a *nondeterministic recurrent neural network* introduced by Siegelmann and Sontag [154, 155]. In their framework, the nondeterminism is expressed via the consideration of an external binary guess stream processed by means of an additional Boolean guess cell.

Formally, a *sigmoidal nondeterministic (first-order) recurrent neural network of type* 1 (denoted by N-RNN) consists of a recurrent neural network $\mathcal{N}$ as described in Section 7.4.1, except that it contains $M+1$ rather than M Boolean input cells $(u_i)_{i=1}^{M+1}$. The Boolean cell u_{M+1}, called the *guess cell*, carries the Boolean source of nondeterminism to be considered [34, 155].

Given some N-RNN $\mathcal{N}$, every element $g = g(0)g(1)g(2)\cdots \in \mathbb{B}^\omega$ is called a *guess stream* for $\mathcal{N}$. Assuming the initial state of the network to be $\langle \boldsymbol{x}(0), \boldsymbol{y}(0)\rangle = \langle \boldsymbol{0}, \boldsymbol{0}\rangle$, any infinite input stream $s = (\boldsymbol{u}(t))_{t\in\mathbb{N}} \in (\mathbb{B}^M)^\omega$ and guess stream $g = (g(t))_{t\in\mathbb{N}} \in \mathbb{B}^\omega$ induce via Equations (7.2) and (7.3) two infinite sequences of states and output states

$$c_{(s,g)} = (\langle \boldsymbol{x}(t), \boldsymbol{y}(t)\rangle)_{t\in\mathbb{N}} \in ([0,1]^N \times \mathbb{B}^P)^\omega$$

$$c'_{(s,g)} = (\boldsymbol{y}(t))_{t\in\mathbb{N}} \in (\mathbb{B}^P)^\omega$$

called the *computation* and *Boolean computation* of $\mathcal{N}$ induced by (s, g), respectively. Furthermore, the Definition of an *attractor* remains unchanged in this case (cf. Section 7.4.1), and we assume that any N-RNN is equipped with a corresponding classification of all of its attractors into meaningful and spurious types. A computation of a N-RNN of type 1 is illustrated in Figure 39.

An infinite input stream $s \in (\mathbb{B}^M)^\omega$ is then called *meaningful* if there exists some guess stream $g \in \mathbb{B}^\omega$ such that $\inf(c'_{(s,g)})$ is a meaningful attractor, and is called *spurious* otherwise, i.e., if for all guess streams $g \in \mathbb{B}^\omega$, the set $\inf(c'_{(s,g)})$ is a spurious attractor. The set of all meaningful input streams is called the *neural ω-language recognized by* $\mathcal{N}$ and is denoted by $L(\mathcal{N})$. A set $L \subseteq (\mathbb{B}^M)^\omega$ is said to be *recognizable* by some nondeterministic recurrent neural network of type 1 if there exists a N-RNN $\mathcal{N}$ such that $L(\mathcal{N}) = L$.

Six different models of N-RNNs can be considered according to the nature of their synaptic weights:

1. the *static rational N-RNNs* (N-St-RNN[$\mathbb{Q}$]s) refer to the class of all N-RNNs whose every weights are static and modelled by rational values.

2. the *static real (or analog) N-RNNs* (N-St-RNN[$\mathbb{R}$]s) refer to the class of all N-RNNs whose every weights are static and modelled by real values.

3. the *bi-valued evolving rational N-RNNs* (N-Ev$_2$-RNN[$\mathbb{Q}$]s) refer to the class of all N-RNNs whose every evolving weights are bi-valued and every static weights are rational.

4. the *bi-valued evolving real N-RNNs* (N-Ev$_2$-RNN[$\mathbb{R}$]s) refer to the class of all N-RNNs whose every evolving weights are bi-valued and every static weights are real.

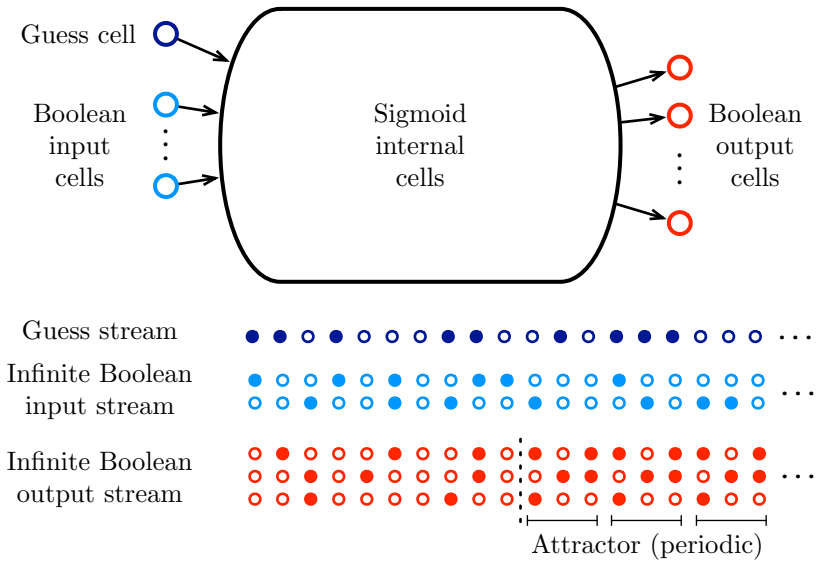


Figure 39 – Illustration of the computational process performed by some N-RNN of type 1. The infinite Boolean input stream $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots \in (\mathbb{B}^M)^\omega$, represented by the blue pattern, together with the guess stream $g = g(0)g(1)g(2)\cdots \in \mathbb{B}^\omega$, represented by the dark-blue pattern, induce a correspond Boolean output stream – or Boolean computation – $c'_s = \boldsymbol{y}(0)\boldsymbol{y}(1)\boldsymbol{y}(2)\cdots \in (\mathbb{B}^P)^\omega$, represented by the red pattern. As in Figure 37, the network necessarily enters into some attractor dynamics.

5. the *(general) evolving rational N-RNNs* (N-Ev-RNN[Q]s) refer to the class of all N-RNNs whose every evolving and static weights are rational.

6. the *(general) evolving real N-RNNs* (N-Ev-RNN[$\mathbb{R}$]s) refer to the class of all N-RNNs whose every evolving and static weights are real.

The following strict inclusions, illustrated in Figure 41, hold by definition:

$$\begin{array}{ccccc} \text{N-St-RNN[Q]s} & \subsetneq & \text{N-Ev}_2\text{-RNN[Q]s} & \subsetneq & \text{N-Ev-RNN[Q]s} \\ \rotatebox{90}{$\subsetneq$} & & \rotatebox{90}{$\subsetneq$} & & \rotatebox{90}{$\subsetneq$} \\ \text{N-St-RNN[}\mathbb{R}\text{]s} & \subsetneq & \text{N-Ev}_2\text{-RNN[}\mathbb{R}\text{]s} & \subsetneq & \text{N-Ev-RNN[}\mathbb{R}\text{]s} \end{array}$$

7.5.2 The Model: Neural Networks of Type 2

We now introduce another notion of *nondeterministic recurrent neural network*, which is, in our sense, closer to the biological framework. The nondeterminism is expressed as a set of possible evolving patterns that the synaptic connections of the network might follow over the successive time steps. At the beginning of a computation, the network selects one such possible evolving pattern – in a nondeterministic manner – and then sticks to it throughout its whole computational process.

Formally, a *sigmoidal nondeterministic (first-order) recurrent neural network of type 2* (denoted by Ñ-RNN) consists of a pair $(\tilde{\mathcal{N}}, E)$, where $\tilde{\mathcal{N}}$ is a recurrent neural network with a dynamics governed by Equations (7.2) and (7.3), and $E \subseteq ([S, S']^K)^\omega$

is a set of infinite sequences of K-dimensional vectors describing the set of all possible evolutions[18] for the K evolving synaptic connections of $\tilde{\mathcal{N}}$ (K is assumed to be smaller than the total number of synaptic connections of $\tilde{\mathcal{N}}$). According to this definition, any D-RNN $\mathcal{N}$ is a particular case of a Ñ-RNN $(\tilde{\mathcal{N}}, E)$ where the evolution set E is reduced to a singleton.

Given some Ñ-RNN $(\tilde{\mathcal{N}}, E)$, every element $e = \boldsymbol{e}(0)\boldsymbol{e}(1)\boldsymbol{e}(2)\cdots \in E$ is called a possible *evolution* for $(\tilde{\mathcal{N}}, E)$, where each vector $\boldsymbol{e}(t)$ describes the values of the K evolving synaptic weights of $\tilde{\mathcal{N}}$ at time step t. Assuming the initial state of the network to be $\langle \boldsymbol{x}(0), \boldsymbol{y}(0)\rangle = \langle \boldsymbol{0}, \boldsymbol{0}\rangle$, any infinite input stream $s = (\boldsymbol{u}(t))_{t\in\mathbb{N}} \in (\mathbb{B}^M)^\omega$ and evolution $e \in E$ induce via Equations (7.2) and (7.3) two infinite sequences of states and output states

$$c_{(s,e)} = (\langle \boldsymbol{x}(t), \boldsymbol{y}(t)\rangle)_{t\in\mathbb{N}} \in ([0,1]^N \times \mathbb{B}^P)^\omega$$
$$c'_{(s,e)} = (\boldsymbol{y}(t))_{t\in\mathbb{N}} \in (\mathbb{B}^P)^\omega$$

called the *computation* and *Boolean computation* of $(\tilde{\mathcal{N}}, E)$ induced by (s, e), respectively. Furthermore, the definition of an *attractor* is once again unchanged and we assume that any Ñ-RNN is equipped with a corresponding classification of all of its attractors into meaningful and spurious types. A computation of a Ñ-RNN of type 2 is illustrated in Figure 40.

An infinite input stream $s \in (\mathbb{B}^M)^\omega$ is called *meaningful* if *there exists* some evolution $e \in E$ such that $\inf(c'_{(s,e)})$ is a meaningful attractor, and it is called *spurious* otherwise, i.e., if *for all* evolutions $e \in E$, the set $\inf(c'_{(s,e)})$ is a spurious attractor. The set of all meaningful input streams is called the *neural ω-language recognized by* $(\tilde{\mathcal{N}}, E)$ and is denoted by $L((\tilde{\mathcal{N}}, E))$. A set $L \subseteq (\mathbb{B}^M)^\omega$ is said to be *recognizable* by some nondeterministic recurrent neural network of type 2 if there exists a Ñ-RNN $(\tilde{\mathcal{N}}, E)$ such that $L((\tilde{\mathcal{N}}, E)) = L$.

Four different models of Ñ-RNNs are considered according to the nature of their synaptic weights.

1. the *bi-valued rational Ñ-RNNs* (Ñ-Ev$_2$-RNN[$\mathbb{Q}$]s) is the class of all Ñ-RNNs whose every evolving weights are bi-valued and every static weights are rational.
2. the *(general) rational Ñ-RNNs* (Ñ-Ev-RNN[$\mathbb{Q}$]s) is the class of all Ñ-RNNs whose every evolving and static weights are rational.
3. the *bi-valued real Ñ-RNNs* (Ñ-Ev$_2$-RNN[$\mathbb{R}$]s) is the class of all Ñ-RNNs whose every evolving weights are bi-valued and every static weights are real.
4. the *(general) real Ñ-RNNs* (Ñ-Ev-RNN[$\mathbb{R}$]s) is the class of all Ñ-RNNs whose every evolving and static weights are real.

The following strict inclusions, illustrated in Figure 41, hold by definition:

$$\begin{array}{ccc} \text{Ñ-Ev}_2\text{-RNN[}\mathbb{Q}\text{]s} & \subsetneq & \text{Ñ-Ev-RNN[}\mathbb{Q}\text{]s} \\ \rotatebox{90}{$\subsetneq$} & & \rotatebox{90}{$\subsetneq$} \\ \text{Ñ-Ev}_2\text{-RNN[}\mathbb{R}\text{]s} & \subsetneq & \text{Ñ-Ev-RNN[}\mathbb{R}\text{]s} \end{array}$$

[18] In this Chapter, an "evolution" is to be understood as a specific pattern of evolvability of the neural architecture, along the lines of Section 5.5.

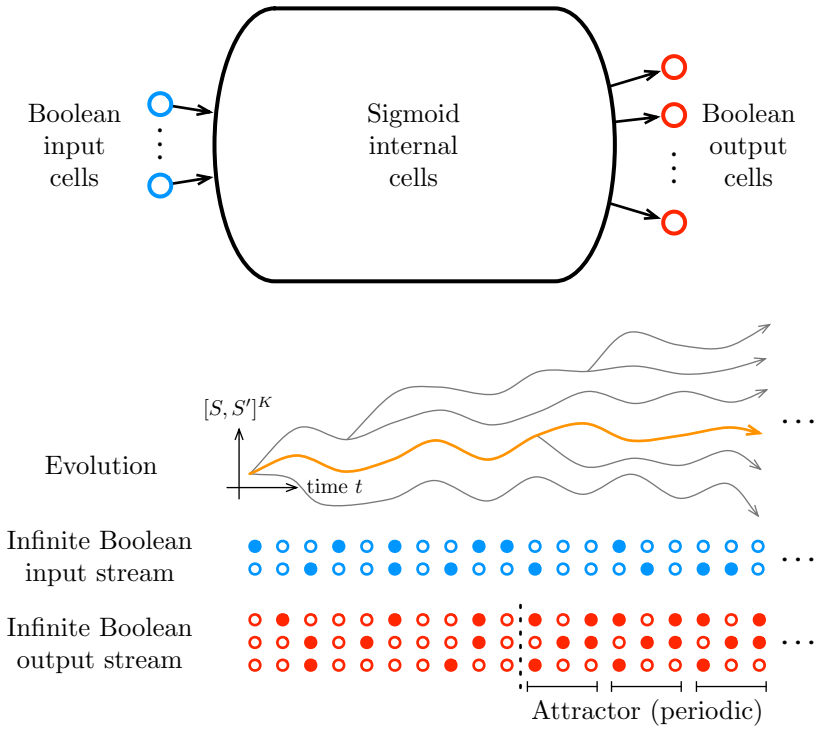


Figure 40 – Illustration of the computational process performed by some ÑN-RNN of type 2. The infinite Boolean input stream $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots \in (\mathbb{B}^M)^\omega$, represented by the blue pattern, together with the evolution $e = \boldsymbol{e}(0)\boldsymbol{e}(1)\boldsymbol{e}(2)\cdots \in \mathbb{B}^\omega$, represented by the orange branch of the grey evolution tree over $([S, S']^K)^\omega$, induce a correspond Boolean output stream – or Boolean computation – $c'_s = \boldsymbol{y}(0)\boldsymbol{y}(1)\boldsymbol{y}(2)\cdots \in (\mathbb{B}^P)^\omega$, represented by the red pattern. As in Figure 37, the network necessarily enters into some attractor dynamics.

Depending on whether $(\tilde{\mathcal{N}}, E)$ is either a ÑN-Ev$_2$-RNN[$\mathbb{Q}$], or a ÑN-Ev$_2$-RNN[$\mathbb{R}$], or a ÑN-Ev-RNN[$\mathbb{Q}$], or a ÑN-Ev-RNN[$\mathbb{R}$], one has either $E \subseteq (\mathbb{B}^K)^\omega$, or $E \subseteq ((\mathbb{Q} \cap [S, S'])^K)^\omega$, or $E \subseteq ((\mathbb{R} \cap [S, S'])^K)^\omega = ([S, S']^K)^\omega$, respectively. Accordingly, we assume from now on that $(\mathbb{Q} \cap [S, S'])^K$ and $(\mathbb{R} \cap [S, S'])^K = [S, S']^K$ are equipped with the subspace topologies[19] of $\mathbb{Q}^K$ and $\mathbb{R}^K$, and that $((\mathbb{Q} \cap [S, S'])^K)^\omega$ and $((\mathbb{R} \cap [S, S'])^K)^\omega$ are equipped with the product topologies of these subspace topologies, respectively. Consequently, the three spaces $(\mathbb{B}^K)^\omega$, $((\mathbb{Q} \cap [S, S'])^K)^\omega$ and $((\mathbb{R} \cap [S, S'])^K)^\omega = ([S, S']^K)^\omega$ are countable products of Polish spaces, and thus are Polish. From this point onwards, we assume that the evolution set E is a closed subset of these Polish subspaces, and hence is also Polish [87].[20] From a biological perspective, this assumption is indeed sensible, since every closed subset of such a product space admits a representation in terms of infinite branches of a given tree [87]; consequently, our evolution sets are naturally identified with 'trees of evolution'.

[19] Given a topological space $(S, \mathcal{T})$ and a subset X of S, the subspace topology on X is defined as $\mathcal{T}_X = \{X \cap U : U \in \mathcal{T}\}$.

[20] The forthcoming results hold equally true even with E taken as $\mathbf{\Pi}^0_2$.

7.5.3 Relationship Between the Two Models

It is worth mentioning that the nondeterminism of type 1 is only a special case of that of type 2. More precisely, for every N-RNN $\mathcal{N}$, there exists a $\tilde{\text{N}}$-RNN $(\tilde{\mathcal{N}}, E)$ such that $L(\mathcal{N}) = L((\tilde{\mathcal{N}}, E))$.

Indeed, let $\mathcal{N}$ be some N-St-RNN[$\mathbb{Q}$] (or some N-St-RNN[$\mathbb{R}$], resp.) with M input cells, 1 guess cell, N internal cell, and M output cells. Now, consider the $\tilde{\text{N}}$-Ev$_2$-RNN[$\mathbb{Q}$] (or the $\tilde{\text{N}}$-Ev$_2$-RNN[$\mathbb{R}$], resp.) $(\tilde{\mathcal{N}}, E)$, where $\tilde{\mathcal{N}}$ is the same network as $\mathcal{N}$, but with the guess cell u_{M+1} being considered as a internal cell x' with an associated evolving bias $c'(t)$, and where the (closed) evolution set associated to this sole evolving weight $c'(t)$ is given by $E = \mathbb{B}^\omega$. The accepting conditions of $\mathcal{N}$ and $(\tilde{\mathcal{N}}, E)$ ensure that $L(\mathcal{N}) = L((\tilde{\mathcal{N}}, E))$.

Similarly, let $\mathcal{N}$ be some N-Ev$_2$-RNN[$\mathbb{Q}$] (or some N-Ev$_2$-RNN[$\mathbb{R}$], or N-Ev-RNN[$\mathbb{Q}$], or N-Ev-RNN[$\mathbb{R}$], resp.) with M input cells, 1 guess cell, N internal cell, M output cells, and K evolving synaptic weights whose evolutions are given by the infinite sequence $e = \boldsymbol{e}(0)\boldsymbol{e}(1)\boldsymbol{e}(2)\cdots \in (\mathbb{B}^K)^\omega$ (or $e \in ((\mathbb{Q} \cap [S, S'])^K)^\omega$, or $e \in ((\mathbb{R} \cap [S, S'])^K)^\omega$, resp.). Now, consider the $\tilde{\text{N}}$-Ev$_2$-RNN[$\mathbb{Q}$] (or the $\tilde{\text{N}}$-Ev$_2$-RNN[$\mathbb{R}$], or the $\tilde{\text{N}}$-Ev-RNN[$\mathbb{Q}$], or the $\tilde{\text{N}}$-Ev-RNN[$\mathbb{R}$], resp.) $(\tilde{\mathcal{N}}, E)$, where $\tilde{\mathcal{N}}$ is the same network as $\mathcal{N}$, but with the guess cell u_{M+1} being considered as a internal cell x' with an associated evolving bias $c'(t)$, and where the (closed) evolution set associated with its $K+1$ evolving weights (the K same ones as $\mathcal{N}$ plus $c'(t)$) is given by $E = \prod_{t\in\mathbb{N}} (\{\boldsymbol{e}(t)\} \times \{0, 1\})$. The accepting conditions of $\mathcal{N}$ and $(\tilde{\mathcal{N}}, E)$ ensure that $L(\mathcal{N}) = L((\tilde{\mathcal{N}}, E))$.

According to these considerations as well as to the inclusion relations mentioned in Sections 7.5.1 and 7.5.2, the relationship between the expressive capabilities of the ten models of nondeterministic neural networks described in Figure 41 obtains.

7.5.4 Expressive Power

We provide a characterization of the expressive powers of the ten models of nondeterministic neural networks. First, we show that N-St-RNN[$\mathbb{Q}$]s are computationally equivalent to nondeterministic Muller Turing machines, and hence, recognize the class of Σ_1^1 (lightface) neural ω-languages. Next, we prove that the nine other models of nondeterministic neural networks are computationally equivalent to each other; they recognize the class of $\mathbf{\Sigma_1^1}$ (boldface) neural ω-languages, and hence, are strictly more powerful than nondeterministic Muller Turing machines. These results are summarized in Table 7 and Figure 41.

The following result states that N-St-RNN[$\mathbb{Q}$]s are computationally equivalent to nondeterministic Muller Turing machines.

Theorem 49. *Let $L \subseteq (\mathbb{B}^M)^\omega$. The following conditions are equivalent:*

1. *L is recognizable by some N-St-RNN[$\mathbb{Q}$];*
2. *L is recognizable by some nondeterministic Muller TM;*
3. *$L \in \Sigma_1^1$ (lightface).*

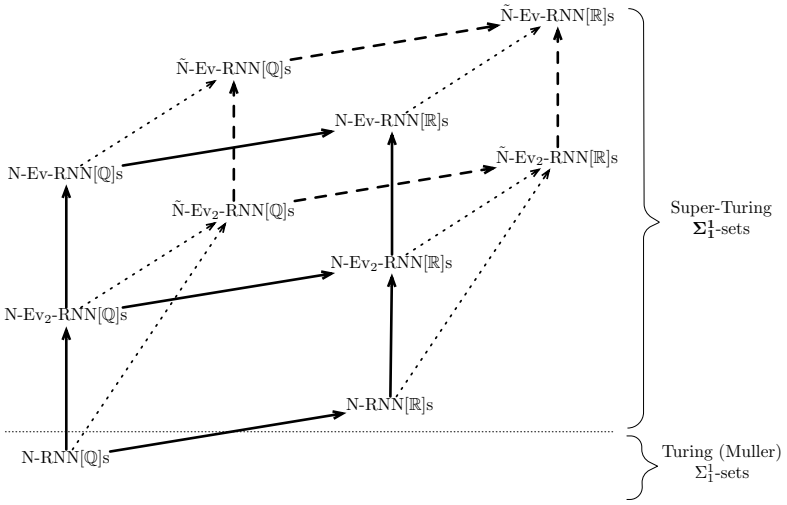


Figure 41 – Relationship between the expressive powers of the ten models of nondeterministic RNNs. There is a directed arrow from one model to the other if the former is less powerful than or equally powerful to the latter. The relation represented by these arrows is clearly transitive. The solid and dashed arrows are given by the inclusion relations established in Sections 7.5.1 and 7.5.2, respectively. The dotted arrows are given by the considerations of Section 7.5.3. In this work, we show that N-St-RNN[ℚ]s are computationally equivalent to nondeterministic Muller Turing machines (Theorem 49), and that the nine other neural models are all equivalent to each other, and strictly more powerful than nondeterministic Muller Turing machines (Theorem 53). The border between the Turing and super-Turing levels is represented by the thin dotted line.

	STATIC	BI-VALUED EVOLVING	GENERAL EVOLVING
ℚ	N-St-RNN[ℚ]s	N-Ev$_2$-RNN[ℚ]s	N-Ev-RNN[ℚ]s
	–	Ñ-Ev$_2$-RNN[ℚ]s	Ñ-Ev-RNN[ℚ]s
	$= \Sigma_1^1$ (lightface)	$= \mathbf{\Sigma_1^1}$ (boldface)	$= \mathbf{\Sigma_1^1}$ (boldface)
	Turing (Muller)	super-Turing	super-Turing
ℝ	N-St-RNN[ℝ]s	N-Ev$_2$-RNN[ℝ]s	N-Ev-RNN[ℝ]s
	–	Ñ-Ev$_2$-RNN[ℝ]s	Ñ-Ev-RNN[ℝ]s
	$= \mathbf{\Sigma_1^1}$ (boldface)	$= \mathbf{\Sigma_1^1}$ (boldface)	$= \mathbf{\Sigma_1^1}$ (boldface)
	super-Turing	super-Turing	super-Turing

Table 7 – Expressive powers of the six models of N-RNNs and the four models of Ñ-RNNs.

Proof. (1) $\leftrightarrow$ (2): By Theorem 45, D-St-RNN[ℚ]s and deterministic Muller TMs are computationally equivalent. This result can be extended to show that N-St-RNN[ℚ]s and nondeterministic Muller TMs are also computationally equivalent. (2) $\leftrightarrow$ (3): This equivalence is given by [159, Theorem 3.5]. □

We now prove that the nine other models of nondeterministic neural networks are computationally equivalent to each other, and strictly more powerful than nondeterministic Muller Turing machines. To begin with, we show that any analytic neural ω-language is recognizable by some N-St-RNN[$\mathbb{R}$] and by some N-Ev$_2$-RNN[$\mathbb{Q}$].

Proposition 50. *Let $L \subseteq (\mathbb{B}^M)^\omega$. If $L \in \mathbf{\Sigma_1^1}$, then L is recognizable by some N-St-RNN[$\mathbb{R}$] and by some N-Ev$_2$-RNN[$\mathbb{Q}$].*

Proof. We first consider the case of a N-St-RNN[$\mathbb{R}$]. Since $L \in \mathbf{\Sigma_1^1}$, there exists some $X \subseteq (\mathbb{B}^M)^\omega \times \{0,1\}^\omega$ such that $X \in \mathbf{\Pi_2^0}$ and $L = \pi_1(X)$ (cf. Chapter 2). Since $X \in \mathbf{\Pi_2^0}$, it can be written as $X = \bigcap_{i\geq 0} \bigcup_{j\geq 0} (p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0,1\}^\omega)$, where each $(p_{i,j}, q_{i,j}) \in (\mathbb{B}^M)^* \times \{0,1\}^*$. Consequently, the set X (and hence also L) is completely determined by the countable sequence of pairs of finite prefixes $((p_{i,j}, q_{i,j}))_{i,j\geq 0}$. Hence, in order to encode the subset L into some real number, it suffices to encode the corresponding sequence of prefixes $((p_{i,j}, q_{i,j}))_{i,j\geq 0}$. We consider some encoding $r_X \in \mathbb{R}$ of the infinite sequence $((p_{i,j}, q_{i,j}))_{i,j\geq 0}$ such that, for any pair of indices $(i,j) \in \mathbb{N} \times \mathbb{N}$, the decoding procedure $(r_X, i, j) \mapsto (p_{i,j}, q_{i,j})$ is recursive.

We now consider the infinite procedure given by Algorithm 6 below. This procedure requires an infinite input stream $s \in (\mathbb{B}^M)^\omega$ and an infinite guess stream $g \in \mathbb{B}^\omega$ provided step by step, as well as the real number r_X. Provided that the real number r_X is given in advance, every instruction of Algorithm 6 is recursive. By construction, Algorithm 6 returns infinitely many 1's on the pair of infinite sequences (s, g) iff (s, g) belongs to X.

Based on the infinite procedure, we provide the description of a N-St-RNN[$\mathbb{R}$] $\mathcal{N}$ such that $L(\mathcal{N}) = L$. First, we consider a neural circuit which stores the incoming values of the input and guess streams $s \in (\mathbb{B}^M)^\omega$ and $g \in \mathbb{B}^\omega$ into $M+1$ designated neurons, and which contains one additional neuron x' with one static real synaptic weight (bias) c' of value r_X. Afterwards, according to the real time computational equivalence between (static) rational RNNs and TMs [155], we consider a (static) rational-weighted RNN which is suitably designed and connected to the above mentioned circuit in order to simulate all the recursive instructions of Algorithm 6. We then add a single Boolean output neuron y and update the whole construction in such a way that y takes an activation value of 1 iff the simulation of Algorithm 6 by our network enters the instruction "returns 1". Finally, the Boolean output cell y leads to the existence of only three possible attractors, namely $\{(0)\}, \{(0),(1)\}$, and $\{(1)\}$. We set $\{(0)\}$ as spurious, and $\{(0),(1)\}$ and $\{(1)\}$ as meaningful. In this way, one has the description of a N-St-RNN[$\mathbb{R}$] $\mathcal{N}$ which suitably simulates the behavior of Algorithm 6. By construction, for any infinite input $s \in (\mathbb{B}^M)^\omega$ and guess $g \in \mathbb{B}^\omega$, the Boolean computation $c'_{(s,g)}$ visits a meaningful attractor iff Algorithm 6 returns infinitely many 1's on the pair of infinite sequences (s, g).

Hence, one has that $s \in L(\mathcal{N})$ iff, by definition, there exists some $g \in \mathbb{B}^\omega$ such that $\inf(c'_{(s,g)})$ is meaningful, iff, by construction, there exists $g \in \mathbb{B}^\omega$ such that Algorithm 6 returns infinitely many 1's on the pair of infinite sequences (s, g), iff, there exists $g \in \mathbb{B}^\omega$ such that the pair $(s, g) \in X$, iff, by definition, $s \in \pi_1(X) = L$.

Algorithm 6 Infinite procedure

Require:

1. Input stream $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots \in (\mathbb{B}^M)^\omega$ supplied step by step at successive time steps $t = 0, 1, 2, \ldots$
2. Guess stream $g = g(0)g(1)g(2)\cdots \in \mathbb{B}^\omega$ supplied step by step at successive time steps $t = 0, 1, 2, \ldots$
3. Real number r_X.

1: **SUBROUTINE 1**
2: $c \leftarrow 0$ // c counts the number of letters provided so far
3: **for all** time step $t \geq 0$ **do**
4: store each incoming Boolean vector $\boldsymbol{u}(t) \in \mathbb{B}^M$
5: store each incoming bit $g(t) \in \{0, 1\}$
6: $c \leftarrow c + 1$
7: **end for**
8: **END SUBROUTINE 1**

9: **SUBROUTINE 2**
10: $i \leftarrow 0, j \leftarrow 0$
11: **loop**
12: wait until $c \geq \max\{|p_{i,j}|, |q_{i,j}|\}$
13: decode $(p_{i,j}, q_{i,j})$ from r_X // recursive procedure if r_X is given
14: **if** $p_{i,j} \subseteq s[0{:}c]$ and $q_{i,j} \subseteq g[0{:}c]$ **then** // $(s,g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0,1\}^\omega$
15: **return** 1 // $\exists\ j$ s.t. $(s,g) \in p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0,1\}^\omega$
16: $i \leftarrow i + 1, j \leftarrow 0$ // test if $(s,g) \in p_{i+1,0} \cdot (\mathbb{B}^M)^\omega \times q_{i+1,0} \cdot \{0,1\}^\omega$
17: **else** // $(s,g) \notin p_{i,j} \cdot (\mathbb{B}^M)^\omega \times q_{i,j} \cdot \{0,1\}^\omega$
18: **return** 0 // $\neg\exists j' \leq j$ s.t. $(s,g) \in p_{i,j'} \cdot (\mathbb{B}^M)^\omega \times q_{i,j'} \cdot \{0,1\}^\omega$
19: $i \leftarrow i, j \leftarrow j + 1$ // test if $(s,g) \in p_{i,j+1} \cdot (\mathbb{B}^M)^\omega \times q_{i,j+1} \cdot \{0,1\}^\omega$
20: **end if**
21: **end loop**
22: **END SUBROUTINE 2**

Therefore, $L(\mathcal{N}) = L$, which shows that L is recognized by the N-St-RNN[$\mathbb{R}$] $\mathcal{N}$.

We now slightly adapt the argument in order to capture the case of a N-Ev$_2$-RNN[Q]. First, we consider an encoding of the infinite sequence $((p_{i,j}, q_{i,j}))_{i,j\geq 0}$ by some infinite word $w_X \in \{0,1\}^\omega$ (rather than by some real number r_X) such that, for any pair of indices $(i,j) \in \mathbb{N} \times \mathbb{N}$, the decoding procedure $(w_X, i, j) \mapsto (p_{i,j}, q_{i,j})$ is recursive. Next, we replace lines 3 and 13 of Algorithm 6 by the following ones:

3′: Auxiliary stream $w_X = w_X(0)w_X(1)w_X(2)\cdots \in \mathbb{B}^\omega$ supplied step by step at successive time steps $t = 0, 1, 2, \ldots$
13′: wait that w_X has become 'long enough' and decode $(p_{i,j}, q_{i,j})$ from w_X

Afterwards, we adapt the construction of the N-St-RNN[$\mathbb{R}$] $\mathcal{N}$ in order to construct a N-Ev$_2$-RNN[Q] $\mathcal{N}'$ which suitably simulates the modified Algorithm 6. In particular, we replace the static real weight c' by the bi-valued evolving weight $c'(t)$ which take as successive values the successive bits of w_X. We update the construction of $\mathcal{N}$ in order to be able to store the successive values of w_X and to decode $(p_{i,j}, q_{i,j})$ from w_X. In this way, one obtains a N-Ev$_2$-RNN[Q] $\mathcal{N}'$ such that $L(\mathcal{N}') = L$, and therefore, L is recognizable by some N-Ev$_2$-RNN[Q]. □

Conversely, we show that every ω-language recognized by some $\tilde{\text{N}}$-Ev-RNN[$\mathbb{R}$] is analytic. Towards this purpose, we need a preliminary Lemma. This statement is based on the important result [26, Lemma 9] which states that, for every $t \geq 0$, any evolving real-weighted neural network $\mathcal{N}$ can be perfectly simulated by some other evolving rational-weighted neural network $\mathcal{N}_t$, up to time step t.

Lemma 51. *Let $(\mathcal{N}, E)$ be some $\tilde{\text{N}}$-Ev-RNN[$\mathbb{R}$] and let $f_{(\mathcal{N},E)} : (\mathbb{B}^M)^\omega \times E \to (\mathbb{B}^P)^\omega$ be the function defined by $f_{(\mathcal{N},E)}(s,e) = c'_{(s,e)}$, where $c'_{(s,e)} = \boldsymbol{y}(0)\boldsymbol{y}(1)\boldsymbol{y}(2)\cdots$ is the Boolean computation produced by $(\mathcal{N}, E)$ when it receives the input stream $s = \boldsymbol{u}(0)\boldsymbol{u}(1)\boldsymbol{u}(2)\cdots$ and selects the evolution $e = \boldsymbol{e}(0)\boldsymbol{e}(1)\boldsymbol{e}(2)\cdots$. Then, $f_{(\mathcal{N},E)}$ is of Baire class* 1.

Proof. Note that the nature of our dynamics ensures that the function $f_{(\mathcal{N},E)}$ is sequential, i.e., for any time step $t \geq 0$, the vectors $\boldsymbol{u}(t)$, $\boldsymbol{e}(t)$ and $\boldsymbol{y}(t)$ are generated simultaneously. Therefore, for any $t \geq 0$, the t first Boolean vectors $\boldsymbol{y}(0)\boldsymbol{y}(1)\cdots\boldsymbol{y}(t-1)$ only depend on the t first input and evolution vectors $\boldsymbol{u}(0)\boldsymbol{u}(1)\cdots\boldsymbol{u}(t-1)$ and $\boldsymbol{e}(0)\boldsymbol{e}(1)\cdots\boldsymbol{e}(t-1)$. Formally, given any basic open set $y \cdot (\mathbb{B}^P)^\omega$ with $y \in (\mathbb{B}^P)^*$, one has that $f^{-1}_{(\mathcal{N},E)}(y \cdot (\mathbb{B}^P)^\omega)$ is of the form

$$\Theta_y = \bigcup_{i \in I} \left[u_i \cdot (\mathbb{B}^M)^\omega \times (e_{\mathbb{R},i} \cdot ([S,S']^K)^\omega \cap E) \right]$$

where each $u_i \in (\mathbb{B}^M)^{|y|}$ and $e_{\mathbb{R},i} \in ([S,S']^K)^{|y|}$ (i.e., u_i and $e_{\mathbb{R},i}$ are sequences of length $|y|$).

Now, the result [26, Lemma 9] ensures that, for any $(s,e) \in (\mathbb{B}^M)^\omega \times E$, the Boolean computation produced by $(\mathcal{N}, E)$ subjected to (s,e) is the very same – up to time step $|y|$ – as that produced by $(\mathcal{N}, E)$ subjected to (s,e'), where e' is obtained by truncating all the components of all the vectors of e after $t(|y|)$ bits[21], for some function $t : \mathbb{N} \to \mathbb{N}$. Formally, by [26, Lemma 9], given any u_i and $e_{\mathbb{R},i}$ as above, there exists a finite sequence $I_{\mathbb{Q},i} = \left(\prod_{k=1}^K [a_{j,k}, b_{j,k}]\right)_{j<|y|} \subseteq ([S,S']^K)^{|y|}$, where each $a_{j,k}, b_{j,k} \in \mathbb{Q}$ and $e_{\mathbb{R},i} \in I_{\mathbb{Q},i}$, and such that

$$f_{(\mathcal{N},E)}\left(u_i \cdot (\mathbb{B}^M)^\omega \times (I_{\mathbb{Q},i} \cdot ([S,S']^K)^\omega \cap E)\right) \subseteq y \cdot (\mathbb{B}^P)^\omega.$$

Hence, if we let $u_i \cdot (\mathbb{B}^M)^\omega \times (I_{\mathbb{Q},i} \cdot ([S,S']^K)^\omega \cap E)$ be denoted by C_i, the above relation is equivalent to $C_i \subseteq f^{-1}_{(\mathcal{N},E)}(y \cdot (\mathbb{B}^P)^\omega) = \Theta_y$, for each $i \in I$, and thus $\bigcup_{i\in I} C_i \subseteq \Theta_y$. On the other hand, by construction of $I_{\mathbb{Q},i}$, one has $\Theta_y \subseteq \bigcup_{i\in I} C_i$. Therefore, $\Theta_y = \bigcup_{i\in I} C_i$, i.e.,

$$\Theta_y = \bigcup_{i \in I} \left[u_i \cdot (\mathbb{B}^M)^\omega \times (I_{\mathbb{Q},i} \cdot ([S,S']^K)^\omega \cap E) \right].$$

Since the bounds of the hyper-intervals involved in the $I_{\mathbb{Q},i}$'s are rational numbers, there exist only countably many distinct u_i and $I_{\mathbb{Q},i}$, and thus, Θ_y can be rewritten as

$$\Theta_y = \bigcup_{i \in J} \left[u_i \cdot (\mathbb{B}^M)^\omega \times (I_{\mathbb{Q},i} \cdot ([S,S']^K)^\omega \cap E) \right]$$

[21] Note that for any $r \in \mathbb{R}$, the set of $r' \in \mathbb{R}$ such that the truncations of (the binary representations of) r and r' after n bits are the same is a closed interval of the form $[a,b]$, where $a, b \in \mathbb{Q}$ and $r \in [a,b]$. Formally, a and b are the rational numbers whose binary representations are $r|_n \cdot 0^\omega$ and $r|_n \cdot 1^\omega$, respectively, where $r|_n$ denotes the truncation of (the binary representations of) r after n bits.

where the index set J is countable.

Now, notice that for each $i \in J$, the sets $u_i \cdot (\mathbb{B}^M)^\omega$ and $I_{\mathbb{Q},i} \cdot ([S, S']^K)^\omega \cap E$ are clopen and closed sets of $(\mathbb{B}^M)^\omega$ and E, respectively. Consequently, $u_i \cdot (\mathbb{B}^M)^\omega \times (e_{\mathbb{R},i} \cdot ([S, S']^K)^\omega \cap E)$ is a closed set of $(\mathbb{B}^M)^\omega \times E$, as a product of closed sets, and Θ_y is a $\mathbf{\Sigma}_2^0$-set of $(\mathbb{B}^M)^\omega \times E$, as a countable union of closed sets. Therefore, $f_{(\mathcal{N},E)}$ is of Baire class 1. □

Proposition 52. *Let $(\mathcal{N}, E)$ be some $\tilde{N}$-Ev-RNN[$\mathbb{R}$]. Then $L((\mathcal{N}, E)) \in \mathbf{\Sigma}_1^1$.*

Proof. Since $\mathcal{N}$ contains finitely many Boolean output cells, it can exhibit finitely many possible output states, and thus also finitely many possible attractors. This feature is independent from the nondeterministic behavior associated with the set of possible evolutions E. Hence, suppose that $\mathcal{N}$ contains the I meaningful attractors $A_i = \{\boldsymbol{b}_{i_1}, \ldots, \boldsymbol{b}_{i_{k(i)}}\}$, for $i = 1, \ldots, I$, where $1 \leq i_1 < \ldots < i_{k(i)} \leq 2^P$, and where $\boldsymbol{b}_n$ denotes the n-th Boolean vector of $\mathbb{B}^P$ according to the lexicographic order. Then, the ω-language $L((\mathcal{N}, E))$ can be expressed by the following sequence of equalities (where we use the fact that $f_{(\mathcal{N},E)}$ is of Baire class 1, established in Lemma 51):

$$
\begin{aligned}
&L((\mathcal{N}, E)) \\
&= \{s \in (\mathbb{B}^M)^\omega : \text{there exists } e \in E \text{ s.t. } \inf(c'_{(s,e)}) \text{ is a meaningful attractor}\} \\
&= \{s \in (\mathbb{B}^M)^\omega : \text{there exists } e \in E \text{ s.t. } \inf(c'_{(s,e)}) = A_i, \text{ for some } i = 1, \ldots, I\} \\
&= \pi_1(\{(s,e) \in (\mathbb{B}^M)^\omega \times E : \inf(c'_{(s,e)}) = A_i, \text{ for some } i = 1, \ldots, I\}) \\
&= \pi_1(\bigcup_{1 \leq i \leq I} \{(s,e) \in (\mathbb{B}^M)^\omega \times E : \inf(c'_{(s,e)}) = A_i\}) \\
&= \pi_1\Big(\bigcup_{1 \leq i \leq I} \{(s,e) \in (\mathbb{B}^M)^\omega \times E : \\
&\qquad \forall j \in \{i_1, \ldots, i_{k(i)}\}, f_{(\mathcal{N},E)}(s,e) \text{ contains infinitely many } \boldsymbol{b}_j\text{'s and} \\
&\qquad \forall j \in \{1, \ldots, 2^P\} \setminus \{i_1, \ldots, i_{k(i)}\}, f_{(\mathcal{N},E)}(s,e) \text{ contains finitely many } \boldsymbol{b}_j\text{'s}\}\Big) \\
&= \pi_1\Big(\bigcup_{1 \leq i \leq I} \Big[\bigcap_{j \in \{i_1, \ldots, i_{k(i)}\}} \{(s,e) \in (\mathbb{B}^M)^\omega \times E : \\
&\qquad\qquad f_{(\mathcal{N},E)}(s,e) \in \underbrace{\bigcap_{n \geq 0} \bigcup_{m \geq 0} (\mathbb{B}^P)^{n+m} \cdot \boldsymbol{b}_j \cdot (\mathbb{B}^P)^\omega}_{\substack{c'_{(s,e)} \text{ contains infinitely many } \boldsymbol{b}_j\text{'s, i.e.} \\ \forall n \geq 0\ \exists m \geq n\ \boldsymbol{y}(n+m) = \boldsymbol{b}_j, \text{ thus in } \mathbf{\Pi}_2^0}} \} \\
&\qquad \cap \bigcap_{\substack{j \in \{1, \ldots, 2^P\} \\ \setminus \{i_1, \ldots, i_{k(i)}\}}} \{(s,e) \in (\mathbb{B}^M)^\omega \times E : \\
&\qquad\qquad f_{(\mathcal{N},E)}(s,e) \in \underbrace{\Big(\bigcap_{n \geq 0} \bigcup_{m \geq 0} (\mathbb{B}^P)^{n+m} \cdot \boldsymbol{b}_j \cdot (\mathbb{B}^P)^\omega\Big)^{\complement}}_{\substack{c'_{(s,e)} \text{ contains only finitely many } \boldsymbol{b}_j\text{'s, i.e.} \\ \text{complement of a } \mathbf{\Pi}_2^0\text{-set, thus in } \mathbf{\Sigma}_2^0}} \}\Big]\Big)
\end{aligned}
$$

$$= \pi_1\Big(\bigcup_{1\le i\le I} \Big[\bigcap_{j\in\{i_1,\ldots,i_{k(i)}\}} \underbrace{f^{-1}_{(\mathcal{N},E)}\Big(\bigcap_{n\ge 0}\bigcup_{m\ge 0} (\mathbb{B}^P)^{n+m}\cdot \boldsymbol{b}_j\cdot(\mathbb{B}^P)^\omega\Big)}_{\substack{\text{preimage by a Baire class 1 function of a } \mathbf{\Pi^0_2}\text{-set}\\ \text{thus in } \mathbf{\Pi^0_3}\ [87]}} \cap$$

$$\bigcap_{\substack{j\in\{1,\ldots,2^P\}\\ \setminus\{i_1,\ldots,i_{k(i)}\}}} \underbrace{f^{-1}_{(\mathcal{N},E)}\Big(\Big(\bigcap_{n\ge 0}\bigcup_{m\ge 0} (\mathbb{B}^P)^{n+m}\cdot \boldsymbol{b}_j\cdot(\mathbb{B}^P)^\omega\Big)^{\complement}\Big)}_{\substack{\text{preimage by a Baire class 1 function of a } \mathbf{\Sigma^0_2}\text{-set}\\ \text{thus in } \mathbf{\Sigma^0_3}\ [87]}} \Big]\Big).$$

It follows that $L((\mathcal{N},E))$ is a projection of a finite union and intersection of $\mathbf{\Pi^0_3}$ and $\mathbf{\Sigma^0_3}$ subsets of the Polish space $(\mathbb{B}^M)^\omega \times E$, and therefore, $L((\mathcal{N},E)) \in \mathbf{\Sigma^1_1}$ (cf. Chapter 2). □

Finally, the following theorem is a direct consequence of previous Propositions 50 and 52.

Theorem 53. *Let $L \subseteq (\mathbb{B}^M)^\omega$. The following conditions are equivalent:*

1. $L \in \mathbf{\Sigma^1_1}$;
2. *L is recognizable by some N-St-RNN[$\mathbb{R}$];*
3. *L is recognizable by some N-Ev$_2$-RNN[$\mathbb{Q}$];*
4. *L is recognizable by some $\tilde{\text{N}}$-Ev$_2$-RNN[$\mathbb{Q}$];*
5. *L is recognizable by some N-Ev-RNN[$\mathbb{Q}$];*
6. *L is recognizable by some $\tilde{\text{N}}$-Ev-RNN[$\mathbb{Q}$];*
7. *L is recognizable by some N-Ev$_2$-RNN[$\mathbb{R}$];*
8. *L is recognizable by some $\tilde{\text{N}}$-Ev$_2$-RNN[$\mathbb{R}$];*
9. *L is recognizable by some N-Ev-RNN[$\mathbb{R}$].*
10. *L is recognizable by some $\tilde{\text{N}}$-Ev-RNN[$\mathbb{R}$].*

Proof. The implications (1) → (2) and (1) → (3) are given by Proposition 50. The implications (2) → (10), (3) → (10), (3) → (4) and (4) → (10), (3) → (5) and (5) → (10), (3) → (6) and (6) → (10), (3) → (7) and (7) → (10), (3) → (8) and (8) → (10), and (3) → (9) and (9) → (10) all hold by the relationships between the various neural models described in Figure 41. The last implication (10) → (1) is provided by Proposition 52. □

7.6 Discussion

We have provided a characterization of the expressive powers of several models of Boolean, sigmoidal deterministic, and sigmoidal nondeterministic first-order recurrent neural networks, in relation with their attractor dynamics.

In the Boolean context (Section 7.3), we have extended, in light of modern automata theory, the seminal equivalence between Boolean recurrent neural networks and finite state automata [89, 118, 119]. We have proven that Boolean recurrent neural networks provided with a less and more general type specification of their attractors are expressively equivalent to Büchi and Muller automata, respectively (Theorems 28 and 37). Consequently, they recognize up to the class of ω-rational neural languages. The Wagner hierarchy [182] can therefore be transposed from the automaton to the neural network context, inducing two decidable hierarchical classifications, and in turn, two novel attractor-based complexity measures for Boolean neural networks (Theorems 32, 33, 41, 42). This complexity is linked to the intricacy of the attractors' structure of the networks, or more precisely, to the maximal number of times that a network might alternate between meaningful and spurious attractors along their possible computations.

In the sigmoidal deterministic context (Section 7.4), we have considered six different models of recurrent neural networks according to whether their synaptic weights are modelled by rational or real numbers, and according to whether the these synaptic weights are either of a static nature, or able to evolve over time among only two possible values, or able to evolve over time among any possible values between two designated bounds. We have shown that D-St-RNN[$\mathbb{Q}$]s are computationally equivalent to deterministic Muller Turing machines (Theorem 45). The five other models of D-Ev$_2$-RNN[$\mathbb{Q}$]s, D-Ev-RNN[$\mathbb{Q}$]s, D-St-RNN[$\mathbb{R}$]s, D-Ev$_2$-RNN[$\mathbb{R}$]s, D-Ev-RNN[$\mathbb{R}$]s are computationally equivalent to each other and strictly more powerful than deterministic Muller Turing machines – with a class of ω-languages equal to $BC(\mathbf{\Pi}_2^0)$ (Theorem 46).

In the sigmoidal nondeterministic context (Section 7.5), two types of nondeterministic neural networks have been considered. In the first case, the nondeterminism is expressed via some external binary guess stream processed by means of an additional Boolean guess cell, along the very lines of that introduced in [154, 155]. In the second case, the nondeterminism is expressed as a set of possible evolving patterns that the synaptic connections of the network might follow over the successive time steps. At the beginning of a computation, the network selects one such possible evolving pattern – in a nondeterministic manner – and then sticks to it throughout its whole computational process. Six models of neural networks of type 1 and four of type 2 have been considered according to the nature of their synaptic weights: rational, real, static, bi-valued evolving or general evolving. It has been observed that the nondeterminism of type 1 is a special case of that of type 2. We have proven that the rational-weighted static neural networks of type 1 are computationally equivalent to the nondeterministic Muller Turing machines, and hence, recognize the class of Σ_1^1 (lightface) neural ω-languages (Theorem 49). The nine other models of neural networks are computationally equivalent to each other; they recognize the class of $\mathbf{\Sigma_1^1}$ (boldface) neural ω-languages, and hence, are strictly more powerful than the nondeterministic Muller Turing machines (Theorem 53).

These results show that nondeterminism injects an extensive amount of computational power – from $BC(\mathbf{\Pi_2^0})$ to $\mathbf{\Sigma_1^1}$ – to the neural systems (see Tables 6 and 7). In the nondeterminism of type 2, as opposed to that of type 1 and to the deterministic case, the consideration of real synaptic weights does actually not add any extra

computational power to the neural networks. The added value of the power of the continuum is somehow absorbed by the nondeterminism, and any kind of analog assumption can therefore be dropped without compromizing the achievement of a maximal computational power.

Overall, these results constitute a precise generalization to the current computational context of those obtained for the cases of classical as well as interactive computations (see Chapters 5 and 6) [21, 24, 25, 26, 32, 35]. In the deterministic as well as in the nondeterministic context, the consideration of bi-valued evolving capabilities provides the possibility to break the Turing barrier of computation and achieve a maximal super-Turing expressive power. The additional consideration of real synaptic weights or of any more general phenomenon of architectural evolvability would actually not increase further the capabilities of the neural networks. As already mentioned, this feature is of specific interest, since discrete architectural evolving phenomena are indeed observable in biological neural networks, as opposed to the power of the continuum, which remains at a conceptual level.

Besides, our attractor-based approach to the computational capabilities of recurrent neural networks is justified by the fact that, in our model, the periodic attractor dynamics of the neural networks are the phenomena that precisely underly the arising of spatiotemporal patterns of discharges [30, 31] – a feature which is assumed to be significantly involved in the processing and coding of information in the brain [2, 3, 4, 5, 80, 116, 137, 174, 175, 178, 180]. This correspondence between *periodic attractors* and *spatiotemporal patters* is illustrated in Figure 26.

In our framework, the computational capabilities of recurrent neural networks is measured by the topological complexity of their underlying neural ω-language, i.e., the set of input streams which induce meaningful attractor dynamics. This correlation between "computational capabilities" and "topological complexity" takes its full meaning once we understand that the latter concept corresponds precisely to the ability of the networks to perform more or less complicated classification tasks of their input streams, via the manifestation of meaningful or spurious attractor dynamics.

To illustrate this feature, consider some recurrent neural network $\mathcal{N}$ whose associated ω-language would be the (very) basic $\mathbf{\Sigma}_1^0$-set $L(\mathcal{N}) = \mathbf{0} \cdot \mathbf{0} \cdot \mathbf{1} \cdot (\mathbb{B}^M)^\omega$ (where $\mathbf{0}$ and $\mathbf{1}$ denote the M-dimensional vectors with only 0's and 1's, respectively). If some input stream s is supplied to the network $\mathcal{N}$, then, in order to correctly classify s with respect to $L(\mathcal{N})$, the network simply needs to scan the three first elements of s: if they correspond to $\mathbf{0}$, $\mathbf{0}$ and $\mathbf{1}$, $\mathcal{N}$ accepts the whole input by entering into some meaningful attractor (since $s \in \mathbf{0} \cdot \mathbf{0} \cdot \mathbf{1} \cdot (\mathbb{B}^M)^\omega$); otherwise, $\mathcal{N}$ rejects the input by entering into some spurious attractor (since $s \notin \mathbf{0} \cdot \mathbf{0} \cdot \mathbf{1} \cdot (\mathbb{B}^M)^\omega$). Consequently, in this basic case, the classification task of $\mathcal{N}$ reduces to the trivial scanning procedure of the three first elements of s in order to accept or reject the input. However, in cases where the ω-language $L(\mathcal{N})$ consists of a more topologically complicated set, the classification task performed by $\mathcal{N}$ to accept or reject the input would amount to a much more intricate "procedure".

In the Boolean and in the sigmoidal deterministic contexts (Sections 7.3 and 7.4), those classification tasks will never be more complicated than the membership problem of a $BC(\mathbf{\Pi}_2^0)$-set. The essence of such a procedure is described in Algo-

rithm 5. In the sigmoidal nondeterministic context (Section 7.5), the complexity of these tasks reduces to the membership problem of a $\mathbf{\Sigma}_1^1$-set. Such a classification procedure is described in Algorithm 6. In this case, the power of the nondeterminism is crucially involved: it provides – through some auxiliary guess stream (for the nondeterminism of type 1) or through the evolvability of the network (for the nondeterminism of type 2) – an additional encoded information which allows the network to reduce its classification task to that of a $\mathbf{\Pi}_2^0$ ω-language only.[22] In all cases, the output of the classification task is provided by an attractor dynamics: a meaningful attractor signifies the acceptance of the input, and a spurious one signifies rejection.

Note that many of these classification tasks are not of an "algorithmic" nature (in the classical sense of the term), since the networks are able to recognize ω-languages that cannot be recognized by any ω-Turing machine. They correspond to some "procedures" lying beyond the reach of the Turing machine model of computation, and accordingly, could be qualified as *super-Turing* or *hypercomputational* procedures.

We can illustrate these theoretical considerations in light of experimental results. For instance, in an experimental paradigm, freely-moving behaving rats were trained to discriminate two classes of human vowels [51, 179]. If the discrimination was correct, the rats had to go to a specific area of the labyrinth and received a food reward. After reaching a steady-state performance, usually above 90%, the rats underwent a brain surgery and multiple electrodes were fit in an area of the cerebral cortex connected to the ascending auditory pathway and in the inferolimbic cortex, that is an area of the cerebral cortex associated with higher cerebral functions (processing of cross-contingencies, time relations, rehearsal of memory traces). During the intracellular recording, several spatiotemporal patterns associated to specific behaviors were found. Figure 42 shows an example of a spatiotemporal pattern of discharges associated to the class of stimuli that were priming the food reward. Note that the firing pattern was associated with the equivalence class of the vowel, irrespective of the pitch or of any other simple acoustical feature.

Here, we are in a case of a biological neural network performing some discrimination tasks among equivalence classes of human vowels. The discrimination of the auditory input is achieved via some specific attractor dynamics of the neural network activity, which, in turn, evokes precise spatiotemporal patterns of discharges that can be detected experimentally. According to our theoretical model, *the neural ω-language* of the neural networks would correspond to the set of auditory inputs which belong to the suitable equivalence class of a human vowels. The *classification task* of the neural network consists in determining whether some given auditory input belongs to its neural ω-language or not. The *output* of this classification process is expressed via some *meaningful or spurious attractor dynamics* which, in turn, evoke precise *spatiotemporal patterns of discharge*. The *computational power* of the neural network is determined by the topological complexity of its underlying ω-language, or equivalently, by the complexity of the classification task associated with this ω-language.

Finally, from a general perspective, our study provides a novel theoretical ap-

[22] This corresponds precisely the idea of the proof of Proposition 50

proach to the crucial role that attractors and spatiotemporal patterns of discharges play in the computational capabilities of neural networks, and more generally, in the processing and coding of information in the brain. They establish a link between the attractor dynamics of the networks, their spatiotemporal patterns of discharge, and their ability to perform more or less intricate classification tasks of their input streams.

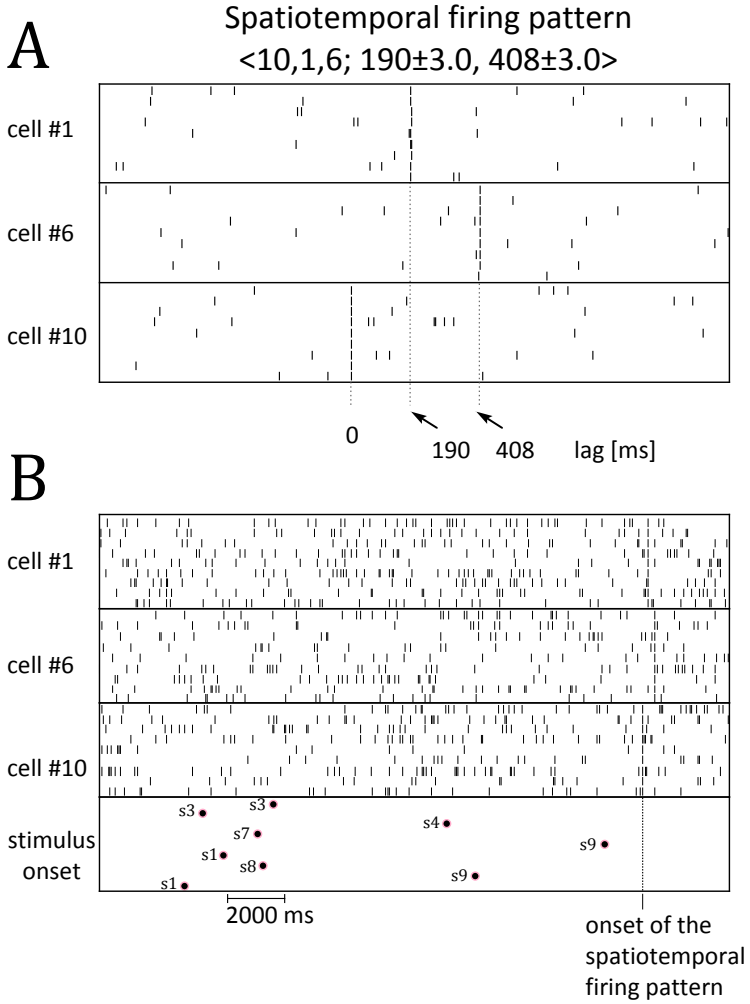


Figure 42 – A. Raster display of the activities of three cortical neurons: Cells #1 and #6 were recorded in the auditory area Te1 from the left hemisphere from two different electrodes, and cell #10 in the auditory area Te3 from the right hemisphere. The rasters are aligned by displaying the first spike in the pattern at lag 0 ms. The pattern occurred 9 times during the session, starting with a spike of cell #10, then after 190 ms a spike of cell #1 with a jitter ± 3 ms and then after 218 ± 3 ms a spike of cell #6. This pattern is denoted $\langle 10, 1, 6; 190 \pm 3, 408 \pm 3 \rangle$. **B.** The same triplet is plotted at a different time scale in order to show the corresponding stimuli onset times. Notice that the pattern occurred always after stimuli of the class 's', but that the lag after specific stimuli, i.e. 's9', is much shorter than after other specific stimuli, i.e. 's1'.

8 Conclusion

In this manuscript, we have reviewed some achievements concerning the computational capabilities of various recurrent neural network models involved in diverse computational contexts: classical, interactive, and attractor-based. Overall, the Boolean static recurrent neural networks are computationally equivalent to finite state automata, irrespective of whether their synaptic weights are modelled by rational or real numbers. The sigmoidal static rational-weighted and real-weighted neural nets are equivalent to Turing machines and Turing machines with advices, respectively. The sigmoidal evolving neural networks are also equivalent to Turing machines with advices – and hence to static real neural nets –, irrespective of whether their synaptic weights are modelled by rational or real numbers, and their patterns of evolvability restricted to binary updates or expressed by any more general form of updating.

In the contexts of classical and interactive computations (Chapters 5 and 6), the results support the already mentioned *Thesis of Analog Computation* [149, 154] and *Thesis of Interactive Computation* [100]. In view of our novel results concerning the capabilities of evolving neural networks, these claims could be extended as follows: *any analog and/or evolving system involved in a classical or interactive paradigm of computation can be captured by the Turing machine with advice model*.

In the three paradigms of computation that have been investigated, the incorporation in a basic neural model of either *analog assumptions* (involving the power of the continnuum), on the one hand, or *evolving capabilities*, on the other hand, represents alternative and equivalent ways towards the achievement of maximal super-Turing computational capabilities. Yet as opposed to the power of the continuum, which is a mathematical concept, the evolving capabilities of the networks are by contrast observable in nature, and therefore, more grounded in the physical world. These consideration support the claim that the general mechanism of evolvability should be critically involved in the computational and dynamical capabilities of biological neural networks, and more generally, in the processing and coding of information in the brain. They provide a theoretical complement to the numerous experimental studies emphasizing the importance of the phenomenon of plasticity in the brain's information processing [1, 48, 69].

More generally, these results show that recurrent neural networks represent a natural model of computation beyond the scope of classical Turing machines [33]. They are sometimes invoked to support the debatable claim that some intrinsic dynamical and computational features of neurobiological systems might fail to be captured by Turing-equivalent neural network models, and accordingly, should lie

beyond the scope of standard artificial models of computation.

In fact, the Turing equivalent neural models can only capture brain-like systems that are discrete, based on bit-calculations, and fixed in their architectures. By contrast, the super-Turing neural models can describe brain-like structures involving continuous levels of chemicals as well as adaptive and evolving architectures. In particular, the static analog and the evolving neural models are capable of apprehending non-linear dynamical properties that are most relevant to brain dynamics, such as rich chaotic behaviors [79, 84, 166, 167, 168], as well as dynamical and idealized chaotic systems [149] – which cannot be described by the universal Turing machine model.

As already mentioned earlier, in the case of evolving neural networks, the achievement of super-Turing potentialities depends on the possibility for "nature" to realize non-recursive patterns of evolvability. Otherwise, the whole process could be simulated by some Turing machine. The assumptions that nature would not only follow preprogrammed patterns, that biological structures could involve non-recursive processes, like purely random phenomena for instance, would suffice to acknowledge the existence of hypercomputational capabilities, for these features cannot be simulated by the Turing machine model. But even with these premises accepted, the issue of the possibility to harness such hypercomputational capabilities remains open, and of paramount importance. For deeper philosophical considerations about hypercomputation, see for instance [46, 47, 126, 160, 161].

In the context of noisy neural networks (not treated in this manuscript), the computational capabilities of the networks rely on the specific conception of noise that we consider [26]. An analog noise would generally decrease the capabilities of the systems to those of finite state automata [18, 111, 115]; by contrast, some discrete source of stochasticity would rather maintain or even increase the capabilities of the networks [153]. The empirical questions concerning the nature of the noise that could occur in biological neural networks and the effects that the noise could have on the functioning of the networks remain open [46]. This issue may not be reduced to the simple dichotomy between either an analog or a discrete nature of noise. For instance, one might consider some source of analog noise itself subjected to fluctuations over time, according to some non-recursive pattern.

More generally, Cicurel and Nicolelis have recently argued that the brain cannot be simulated by any Turing machine [43]. They claim that the brain should rather be conceived as a *hybrid digital-analog computational engine* (HDACE):

> According to the relativistic brain theory, complex central nervous systems like ours generate, process, and store information through the recursive interaction of a hybrid digital-analog computation engine (HDACE). In the HDACE, the digital component is defined by the spikes produced by neural networks distributed all over the brain, whereas the analog component is represented by the superimposition of time-varying, neuronal electromagnetic fields (NEMFs), generated by the flow of neuronal electrical signals through the multitude of local and distributed loops of white matter that exist in the mammalian brain. [43, p.27]

Some of our results might provide a theoretical foundation towards the under-

standing of the Turing and super-Turing computational capabilities of such hybrid digital/analog – and also evolving – brain-like models of computation.

Alan Turing himself explained that his machine model is different from the human brain: "Electronic computers are intended to carry out any definite rule of thumb process which could have been done by a human operator working in a disciplined but unintelligent manner" [172]. Yet, he trusted that other models will exist that will describe intelligence better: "My contention is that machines can be constructed which will simulate the behaviour of the human mind very closely" [170]. In 1952, Turing suggested a particular direction, based on adaptability and learning: "If the machine is built to be treated only as a domestic pet, and is spoon-fed with particular problems, it will not be able to learn in the varying way in which human beings learn" [173]. While Turing died within two years and did not manage to realize his own direction, we hope that our results shall constitute a step forward in the study of more intelligent systems, following Turing 1952's call.

For future work, the study of the computational capabilities of more biologically-oriented neural models involved in more bio-inspired paradigms of computation is expected to be pursued. In particular, the computational capabilities of neural networks that are at stake in the crucial mechanism of Spike Timing Dependent Plasticity (STDP) are expected to be investigated. The relationship between the graph topology of the networks and their computational complexity is also of specific interest. Furthermore, we would also be highly interested in the possibility of obtaining some classifications of the super-Turing computational and expressive powers of analog and/or evolving recurrent neural networks, according to some notion of complexity. For instance, Balcázar, Gavaldà and Siegelmann provided a transfinite hierarchization of the super-Turing power of analog neural nets in terms of the Kolomogorov complexity of their real synaptic weights [17]. It is easy to notice that this classification does not hold anymore in the context of evolving neural nets: indeed, evolving networks with real weights of minimal Kolomogorov complexity – i.e., rational weights – can nevertheless achieve maximal super-Turing capabilities. Consequently, the analog and evolving neural networks are, in light of the Kolmogorov complexity of their synaptic weights, not computationally equivalent anymore. This research direction might therefore provide a better understanding of the possible equivalence and discrepancies that exist between the models of analog and evolving networks.

Finally, we hope that such comparative studies between the computational capabilities of neural models and abstract machines might eventually bring further insight to the understanding of both biological and artificial intelligences. We believe that similarly to the foundational work from Turing [171], which played a crucial role in the practical realization of modern computers, further theoretical considerations about neural- and natural-based models of computation shall contribute to the emergence of novel computational technologies, and step by step, open the way to the next computational generation.

BIBLIOGRAPHY

[1] Larry F. Abbott and Sacha B. Nelson. "Synaptic plasticity: taming the beast". In: *Nat. Neurosci.* 3 Suppl. (2000), pp. 1178–1183.

[2] Moshe Abeles. *Corticonics: Neuronal Circuits of the Cerebral Cortex*. 1st. Cambridge University Press, 1991.

[3] Moshe Abeles. *Local Cortical Circuits. An Electrophysiological Study*. Vol. 6. Studies of Brain Function. Berlin Heidelberg New York: Springer-Verlag, 1982.

[4] Moshe Abeles. "Time Is Precious". In: *Science* 304.5670 (2004), pp. 523–524.

[5] M. Abeles et al. "Spatiotemporal firing patterns in the frontal cortex of behaving monkeys". In: *J. Neurophysiol.* 70.4 (1993), pp. 1629–1638.

[6] S.J. van Albada and P.A. Robinson. "Mean-field modeling of the basal ganglia-thalamocortical system. I: Firing rates in healthy and parkinsonian states". In: *J. Theor. Biol.* 257.4 (2009), pp. 642–663.

[7] S.J. van Albada et al. "Mean-field modeling of the basal ganglia-thalamocortical system. II: Dynamics of parkinsonian oscillations". In: *J. Theor. Biol.* 257.4 (2009), pp. 664–688.

[8] Garrett E. Alexander, Michael D. Crutcher, and Mahlon R. DeLong. "Chapter 6. Basal ganglia-thalamocortical circuits: Parallel substrates for motor, oculomotor, prefrontal and limbic functions". In: *The Prefrontal Cortex: Its Structure, Function and Cortex Pathology*. Ed. by H.B.M. Uylings et al. Vol. 85. Progress in Brain Research. Elsevier, 1991, pp. 119–146.

[9] G.E. Alexander and M.D. Crutcher. "Functional architecture of basal ganglia circuits: neural substrates of parallel processing". In: *Trends Neurosci.* 13.7 (1990), pp. 266–271.

[10] Noga Alon, Alexander K. Dewdney, and Teunis J. Ott. "Efficient Simulation of Finite Automata by Neural Nets". In: *J. ACM* 38.2 (1991), pp. 495–514.

[11] René Alquézar and Alberto Sanfeliu. "An Algebraic Framework to Represent Finite State Machines in Single-Layer Recurrent Neural Networks". In: *Neural Computation* 7.5 (1995), pp. 931–949.

[12] Daniel J. Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge University Press, 1992.

[13] D.J. Amit. *Modeling brain function: the world of attractor neural networks*. Cambridge University Press, 1989.

[14] Y. Asai and A.E.P. Villa. “Integration and transmission of distributed deterministic neural activity in feed-forward networks”. In: *Brain Res.* 1434 (2012), pp. 17–33.

[15] Yoshiyuki Asai and Alessandro E.P. Villa. “Reconstruction of underlying nonlinear deterministic dynamics embedded in noisy spike trains”. In: *J. Biol. Phys.* 34.3-4 (2008), pp. 325–340.

[16] Jose L. Balcazar, Josep Diaz, and Joaquim Gabarro. *Structural Complexity I*. 2nd. Springer Publishing Company, Incorporated, 2012.

[17] José L. Balcázar, Ricard Gavaldà, and Hava T. Siegelmann. “Computational power of neural networks: a characterization in terms of Kolmogorov complexity”. In: *IEEE Transactions on Information Theory* 43.4 (1997), pp. 1175–1183.

[18] Asa Ben-Hur, Alexander Roitershtein, and Hava T. Siegelmann. “On probabilistic analog automata”. In: *Theor. Comput. Sci.* 320.2-3 (2004), pp. 449–464.

[19] Olivier Bournez and Michel Cosnard. “On the computational power of dynamical systems and hybrid systems”. In: *Theoretical Computer Science* 168.2 (1996), pp. 417–459.

[20] Daniel A. Butts et al. “Temporal precision in the neural code and the timescales of natural vision”. In: *Nature* 449.7158 (2007), pp. 92–95.

[21] Jérémie Cabessa. “Interactive Evolving Recurrent Neural Networks Are Super-Turing”. In: *Proceedings of ICAART*. Ed. by Joaquim Filipe and Ana L. N. Fred. SciTePress, 2012, pp. 328–333.

[22] Jérémie Cabessa and Jacques Duparc. “Expressive Power of Non-deterministic Evolving Recurrent Neural Networks in Terms of Their Attractor Dynamics”. In: *Unconventional Computation and Natural Computation - 14th International Conference, UCNC 2015, Auckland, New Zealand, August 30 - September 3, 2015, Proceedings*. Ed. by Cristian S. Calude and Michael J. Dinneen. Vol. 9252. Lecture Notes in Computer Science. Springer, 2015, pp. 144–156.

[23] Jérémie Cabessa and Jacques Duparc. “Expressive Power of Nondeterministic Recurrent Neural Networks in Terms of their Attractor Dynamics”. In: *IJUC* (2016, Accepted).

[24] Jérémie Cabessa and Hava T. Siegelmann. “Evolving recurrent neural networks are super-Turing”. In: *Proceedings of IJCNN 2011*. IEEE, 2011, pp. 3200–3206.

[25] Jérémie Cabessa and Hava T. Siegelmann. “The Computational Power of Interactive Recurrent Neural Networks”. In: *Neural Computation* 24.4 (2012), pp. 996–1019.

[26] Jérémie Cabessa and Hava T. Siegelmann. “The Super-Turing Computational Power of plastic Recurrent Neural Networks”. In: *Int. J. Neural Syst.* 24.8 (2014).

[27] Jérémie Cabessa and Alessandro E. P. Villa. “A Hierarchical Classification of First-Order Recurrent Neural Networks”. In: *Chinese Journal of Physiology* 53 (2010), pp. 407–416.

[28] Jérémie Cabessa and Alessandro E. P. Villa. "A Hierarchical Classification of First-Order Recurrent Neural Networks". In: *Proceedings of LATA 2010*. Ed. by Adrian Horia Dediu et al. Vol. 6031. Lecture Notes in Computer Science. Springer, 2010, pp. 142–153.

[29] Jérémie Cabessa and Alessandro E. P. Villa. "An Attractor-Based Complexity Measurement for Boolean Recurrent Neural Networks". In: *PLoS ONE* 9.4 (2014), e94204+.

[30] Jérémie Cabessa and Alessandro E. P. Villa. "Computational capabilities of recurrent neural networks based on their attractor dynamics". In: *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*. IEEE, 2015, pp. 1–8.

[31] Jérémie Cabessa and Alessandro E. P. Villa. "Expressive Power of First-Order Recurrent Neural Networks Determined by their Attractor Dynamics". In: *J. Comput. System Sci.* (2015, Accepted).

[32] Jérémie Cabessa and Alessandro E. P. Villa. "Interactive Evolving Recurrent Neural Networks Are Super-Turing Universal". In: *Proceedings of ICANN 2014*. Ed. by Stefan Wermter et al. Vol. 8681. Lecture Notes in Computer Science. Springer, 2014, pp. 57–64.

[33] Jérémie Cabessa and Alessandro E. P. Villa. "Recurrent Neural Networks – A Natural Model of Computation beyond the Turing Limits". In: *Proceedings of IJCCI 2012*. Ed. by Agostinho C. Rosa et al. SciTePress, 2012, pp. 594–599.

[34] Jérémie Cabessa and Alessandro E. P. Villa. "The expressive power of analog recurrent neural networks on infinite input streams". In: *Theor. Comput. Sci.* 436 (2012), pp. 23–34.

[35] Jérémie Cabessa and Alessandro E. P. Villa. "The Super-Turing Computational Power of Interactive Evolving Recurrent Neural Networks". In: *Proceedings of ICANN 2013*. Ed. by Valeri Mladenov et al. Vol. 8131. Lecture Notes in Computer Science. Springer, 2013, pp. 58–65.

[36] Jérémie Cabessa and Alessandro E.P. Villa. "Artificial Neural Networks: Methods and Applications in Bio-/Neuroinformatics". In: ed. by Petia Koprinkova-Hristova, Valeri Mladenov, and K. Nikola Kasabov. Springer International Publishing, 2015. Chap. Recurrent Neural Networks and Super-Turing Interactive Computation, pp. 1–29.

[37] Cristian Calude and Gheorghe Paun. "Bio-Steps Beyond Turing". In: *BioSystems* 77 (2004), pp. 175–194.

[38] Natalia Caporale and Yang Dan. "Spike timing-dependent plasticity: a Hebbian learning rule". In: *Annu. Rev. Neurosci.* 31 (2008), pp. 25–46.

[39] A. Celletti and A.E.P. Villa. "Determination of Chaotic Attractors in the Rat Brain". In: *Journal of Statistical Physics* 84.5 (1996), pp. 1379–1385.

[40] Alessandra Celletti and Alessandro E.P. Villa. "Low-dimensional chaotic attractors in the rat brain". In: *Biol. Cybern.* 74.5 (1996), pp. 387–393.

[41] G. Chechik, I. Meilijson, and E. Ruppin. "Neuronal regulation: A mechanism for synaptic pruning during brain maturation". In: *Neural Comput.* 11 (1999), pp. 2061–2080.

[42] Gal Chechik. “Spike-Timing-Dependent Plasticity and Relevant Mutual Information Maximization”. In: *Neural Computation* 15.7 (2003), pp. 1481–1510.

[43] Ronald Cicurel and Miguel A. L. Nicolelis. *The Relativistic Brain: How it works and why it cannot be simulated by a Turing machine*. Kios Press, 2015.

[44] Axel Cleeremans, David Servan-Schreiber, and James L. McClelland. “Finite State Automata and Simple Recurrent Networks”. In: *Neural Computation* 1.3 (1989), pp. 372–381.

[45] Ton Coolen and David Sherrington. In: *Mathematical Approaches to Neural Networks*. Ed. by J.G. Taylor. Vol. 51. North-Holland Mathematical Library. Elsevier, 1993, pp. 293–306.

[46] B. Jack Copeland. “Hypercomputation”. In: *Minds Mach.* 12.4 (2002), pp. 461–502.

[47] B. Jack Copeland. “Hypercomputation: philosophical issues”. In: *Theor. Comput. Sci.* 317.1-3 (2004), pp. 251–267.

[48] A. Destexhe and E. Marder. “Plasticity in single neuron and circuit computations”. In: *Nature* 431 (2004).

[49] C. Eliasmith. “A unified approach to building and controlling spiking attractor networks”. In: *Neural Comput.* 17.6 (2005), pp. 1276–1314.

[50] Jeffrey L. Elman. “Finding Structure in Time”. In: *Cognitive Science* 14.2 (1990), pp. 179–211.

[51] Jan L. Eriksson and Alessandro E. P. Villa. “Learning of auditory equivalence classes for vowels by rats”. In: *Behav. Proc.* 73 (2006), pp. 348–359.

[52] Stanley Franklin and Max Garzon. “Neural Computability”. In: *Progress in Neural Networks*. Ed. by Omid Omidvar. Norwood, NJ, USA: Ablex, 1989, pp. 128–144.

[53] Paolo Frasconi et al. “Representation of Finite State Automata in Recurrent Radial Basis Function Networks”. In: *Machine Learning* 23.1 (1996), pp. 5–32.

[54] Paolo Frasconi et al. “Unified Integration of Explicit Knowledge and Learning by Example in Recurrent Networks”. In: *IEEE Trans. Knowl. Data Eng.* 7.2 (1995), pp. 340–346.

[55] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics* 36.4 (1980), pp. 193–202.

[56] Max Garzon and Stanley Franklin. “Neural Computability II”. In: *Proceedings of the Third International Joint Conference on Neural Networks*. Ed. by Omid Omidvar. IEEE, 1989, pp. 631–637.

[57] George L. Gerstein and Kyle L. Kirkland. “Neural assemblies: technical issues, analysis, and modeling”. In: *Neural Networks* 14.6-7 (2001), pp. 589–598.

[58] Marian Gheorghe and Mike Stannett. “Membrane system models for super-Turing paradigms”. In: *Natural Computing* 11.2 (2012), pp. 253–259.

[59] C. Lee Giles et al. “Learning and Extracting Finite State Automata with Second-Order Recurrent Neural Networks”. In: *Neural Computation* 4.3 (1992), pp. 393–405.

[60] Dina Goldin. “Persistent Turing Machines as a Model of Interactive Computation”. In: *Foundations of Information and Knowledge Systems*. Ed. by Klaus-Dieter Schewe and Bernhard Thalheim. Vol. 1762. LNCS. Springer Berlin / Heidelberg, 2000, pp. 116–135.

[61] Dina Goldin, Scott A. Smolka, and Peter Wegner. *Interactive Computation: The New Paradigm*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[62] Dina Goldin and Peter Wegner. “Principles of Interactive Computation”. In: *Interactive Computation*. Ed. by Dina Goldin, Scott A. Smolka, and Peter Wegner. Springer Berlin Heidelberg, 2006, pp. 25–37.

[63] Dina Goldin and Peter Wegner. “The Church-Turing Thesis: Breaking the Myth”. In: *New Computational Paradigms*. Ed. by S. Barry Cooper, Benedikt Löwe, and Leen Torenvliet. Vol. 3526. LNCS. Springer Berlin / Heidelberg, 2005, pp. 152–168.

[64] Dina Goldin and Peter Wegner. “The Interactive Nature of Computing: Refuting the Strong Church-Turing Thesis”. In: *Minds Mach.* 18 (1 2008), pp. 17–38.

[65] Dina Goldin et al. “Turing machines, transition systems, and interaction”. In: *Inf. Comput.* 194 (2 2004), pp. 101–128.

[66] Mark W. Goudreau et al. “First-order versus second-order single-layer recurrent neural networks”. In: *IEEE Transactions on Neural Networks* 5.3 (1994), pp. 511–513.

[67] Ralph Hartley and Harold Szu. “A Comparison of the Computational Power of Neural Network Models”. In: *Proceedings of the IEEE First International Conference on Neural Networks*. Ed. by Charles Butler. IEEE, 1987, pp. 17–22.

[68] Donald O. Hebb. *The organization of behavior : a neuropsychological theory*. John Wiley & Sons Inc., 1949.

[69] A. Holtmaat and K. Svoboda. “Experience-dependent structural synaptic plasticity in the mammalian brain”. In: *Nat. Rev. Neurosci.* 10.9 (2009), pp. 647–658.

[70] J.E. Hoover and P.L. Strick. “Multiple output channels in the basal ganglia”. In: *Science* 259.5096 (1993), pp. 819–821.

[71] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[72] John J. Hopfield. “Neural Networks and Physical Systems with Emergent Collective Computational Abilities”. In: *Proceedings of the National Academy of Sciences* 79 (1982), pp. 2554–2558.

[73] Bill G. Horne and Don R. Hush. “Bounds on the complexity of recurrent neural network implementations of finite state machines”. In: *Neural Networks* 9.2 (1996), pp. 243–252.

[74] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257.

[75] Kurt Hornik, Maxwell Stinchcombe, and Halber White. "Multilayer Feedforward Networks are Universal Approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[76] Heikki Hyötyniemi. "Turing machines are recurrent neural networks". In: *STeP '96 - Genes, Nets and Symbols; Finnish Artificial Intelligence Conference, Vaasa 20-23 Aug. 1996*. Ed. by J. Alander, T. Honkela, and Jakobsson M. STeP '96 - Genes, Nets and Symbols; Finnish Artificial Intelligence Conference, Vaasa, Finland, 20-23 August 1996. Vaasa, Finland: University of Vaasa, Finnish Artificial Intelligence Society (FAIS), 1996, pp. 13–24.

[77] Javier Iglesias, Olga K. Chibirova, and Alessandro E. P. Villa. "Nonlinear Dynamics Emerging in Large Scale Neural Networks with Ontogenetic and Epigenetic Processes". In: *Lecture Notes in Computer Science* 4668 (2007), pp. 579–588.

[78] Javier Iglesias and Alessandro E.P. Villa. "Recurrent spatiotemporal firing patterns in large spiking neural networks with ontogenetic and epigenetic processes". In: *J. Physiol. Paris* 104.3-4 (2010), pp. 137–146.

[79] Kensuke Ikeda. "Chaotic Itinerancy". In: *International Symposia on Information Sciences (ISKIT) '92* (1992). Kyoto Univ.

[80] Yuji Ikegaya et al. "Synfire Chains and Cortical Songs: Temporal Modules of Cortical Activity". In: *Science* 304.5670 (2004), pp. 559–564.

[81] G.M. Innocenti and D.J. Price. "Exuberance in the development of cortical networks". In: *Nature Rev. Neurosci.* 6 (2005), pp. 955–965.

[82] Mihai Ionescu, Gheorghe Păun, and Takashi Yokomori. "Spiking Neural P Systems". In: *Fundam. Inform.* 71.2-3 (2006), pp. 279–308.

[83] D. Joel and I. Weiner. "The organization of the basal ganglia-thalamocortical circuits: Open interconnected rather than closed segregated". In: *Neuroscience* 63.2 (1994), pp. 363–379.

[84] Kunihiko Kaneko and Ichiro Tsuda. "Chaotic itinerancy". In: *Chaos* 13.3 (2003), pp. 926–936.

[85] Nikola Kasabov. *Evolving connectionist systems – the knowledge engineering approach (2. ed.)*. Springer Verlag, 2007, pp. I–XXI, 1–457.

[86] Stuart A Kauffman. *The origins of order: Self-organization and selection in evolution*. New York: Oxford University Press, 1993.

[87] Alexander S. Kechris. *Classical descriptive set theory*. Vol. 156. Graduate Texts in Mathematics. New York: Springer-Verlag, 1995, pp. xviii+402.

[88] Joe Kilian and Hava T. Siegelmann. "The dynamic universality of sigmoidal neural networks". In: *Inf. Comput.* 128.1 (1996), pp. 48–56.

[89] Stephen C. Kleene. "Representation of events in nerve nets and finite automata". In: *Automata Studies*. Ed. by Claude Shannon and John McCarthy. Princeton, NJ: Princeton University Press, 1956, pp. 3–41.

[90] Teuvo Kohonen. *Self-Organization and Associative Memory*. First. Springer, 1984.

[91] Teuvo Kohonen. “Self-organized formation of topologically correct feature maps”. In: *Biological Cybernetics* 43.1 (1982), pp. 59–69.

[92] Stefan C. Kremer. “On the computational power of Elman-style recurrent networks”. In: *Neural Networks, IEEE Transactions on* 6.4 (1995), pp. 1000–1004.

[93] Juri D. Kropotov and Susan C. Etlinger. “Selection of actions in the basal ganglia thalamocortical circuits: review and model”. In: *International Journal of Psychophysiology* 31.3 (1999), pp. 197–217.

[94] A. Leblois et al. “Competition between feedback loops underlies normal and pathological dynamics in the basal ganglia”. In: *J. Neurosci.* 26.13 (2006), pp. 3567–3583.

[95] Jan van Leeuwen and Jiřì Wiedermann. “A Theory of Interactive Computation”. In: *Interactive Computation*. Ed. by Dina Goldin, Scott A. Smolka, and Peter Wegner. Springer Berlin Heidelberg, 2006, pp. 119–142.

[96] Jan van Leeuwen and Jiřì Wiedermann. “Beyond the Turing Limit: Evolving Interactive Systems”. In: *SOFSEM 2001: Theory and Practice of Informatics*. Ed. by Leszek Pacholski and Peter Ružicka. Vol. 2234. LNCS. Springer Berlin / Heidelberg, 2001, pp. 90–109.

[97] Jan van Leeuwen and Jiřì Wiedermann. “How We Think of Computing Today”. In: *Logic and Theory of Algorithms*. Ed. by Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe. Vol. 5028. LNCS. Springer Berlin / Heidelberg, 2008, pp. 579–593.

[98] Jan van Leeuwen and Jiřì Wiedermann. “On Algorithms and Interaction”. In: *Mathematical Foundations of Computer Science 2000*. Ed. by Mogens Nielsen and Branislav Rovan. Vol. 1893. LNCS. Springer Berlin / Heidelberg, 2000, pp. 99–113.

[99] Jan van Leeuwen and Jiřì Wiedermann. “The emergent computational potential of evolving artificial living systems”. In: *AI Commun.* 15 (4 2002), pp. 205–215.

[100] Jan van Leeuwen and Jiřì Wiedermann. “The Turing Machine Paradigm in Contemporary Computing”. In: *Mathematics Unlimited - 2001 and Beyond*. Ed. by Björn Engquist and Wilfried Schmid. LNCS. Springer-Verlag, 2001, pp. 1139–1155.

[101] William A. Little. “The existence of persistent states in the brain”. In: *Mathematical biosciences* 19 (1974), pp. 101–120.

[102] William A. Little and Gordon L. Shaw. “Analytical study of the memory storage capacity of a neural network”. In: *Mathematical biosciences* 39 (1978), pp. 281–290.

[103] Wolfgang Maass. “Fast Sigmoidal Networks via Spiking Neurons”. In: *Neural Computation* 9.2 (1997), pp. 279–304.

[104] Wolfgang Maass. “Lower Bounds for the Computational Power of Networks of Spiking Neurons”. In: *Neural Computation* 8.1 (1996), pp. 1–40.

[105] Wolfgang Maass. "Models for Fast Analog Computation with Spiking Neurons". In: *The Fifth International Conference on Neural Information Processing, ICONIP'R Conference, Kitakyushu, Japan, October 21-23, 1998, Proceedings*. Ed. by Shiro Usui and Takashi Omori. IOA Press, 1998, pp. 187–188.

[106] Wolfgang Maass. "Networks of spiking neurons: The third generation of neural network models". In: *Neural Networks* 10.9 (1997), pp. 1659–1671.

[107] Wolfgang Maass. "Noisy Spiking Neurons with Temporal Coding have more Computational Power than Sigmoidal Neurons". In: *Advances in Neural Information Processing Systems 9, NIPS Conference, Denver, CO, USA, December 2-5, 1996*. Ed. by Michael Mozer, Michael I. Jordan, and Thomas Petsche. MIT Press, 1996, pp. 211–217.

[108] Wolfgang Maass. "On the Computational Complexity of Networks of Spiking Neurons". In: *Advances in Neural Information Processing Systems 7, NIPS Conference, Denver, CO, USA, 1994*. Ed. by Gerald Tesauro, David S. Touretzky, and Todd K. Leen. MIT Press, 1994, pp. 183–190.

[109] Wolfgang Maass. "On the Computational Power of Noisy Spiking Neurons". In: *Advances in Neural Information Processing Systems 8, NIPS Conference, Denver, CO, November 27-30, 1995*. Ed. by David S. Touretzky, Michael Mozer, and Michael E. Hasselmo. MIT Press, 1995, pp. 211–217.

[110] Wolfgang Maass and Henry Markram. "On the computational power of circuits of spiking neurons". In: *J. Comput. Syst. Sci.* 69.4 (2004), pp. 593–616.

[111] Wolfgang Maass and Pekka Orponen. "On the effect of analog noise in discrete-time analog computations". In: *Neural Comput.* 10.5 (1998), pp. 1071–1095.

[112] Wolfgang Maass and Berthold Ruf. "On Computations with Pulses". In: *Inf. Comput.* 148.2 (1999), pp. 202–218.

[113] Wolfgang Maass and Michael Schmitt. "On the Complexity of Learning for a Spiking Neuron (Extended Abstract)". In: *Proceedings of the Tenth Annual Conference on Computational Learning Theory, COLT Conference, Nashville, TN, USA, July 6-9, 1997*. Ed. by Yoav Freund and Robert E. Schapire. ACM, 1997, pp. 54–61.

[114] Wolfgang Maass and Michael Schmitt. "On the Complexity of Learning for Spiking Neurons with Temporal Coding". In: *Inf. Comput.* 153.1 (1999), pp. 26–46.

[115] Wolfgang Maass and Eduardo D. Sontag. "Analog neural nets with Gaussian or other common noise distributions cannot recognize arbitary regular languages". In: *Neural Comput.* 11.3 (1999), pp. 771–782.

[116] Z.F. Mainen and T.J. Sejnowski. "Reliability of spike timing in neocortical neurons". In: *Science* 268.5216 (1995), pp. 1503–1506.

[117] S. J. Martin, P. D. Grimwood, and R. G. M. Morris. "Synaptic Plasticity and Memory: An Evaluation of the Hypothesis". In: *Annu. Rev. Neurosci.* 23.1 (2000), pp. 649–711.

[118] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *Bulletin of Mathematical Biophysic* 5 (1943), pp. 115–133.

[119] Marvin L. Minsky. *Computation: finite and infinite machines*. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1967.

[120] Marvin L. Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[121] H. Nakahara, S. Amari Si, and O. Hikosaka. "Self-organization in the basal ganglia with modulation of reinforcement signals". In: *Neural Comput.* 14.4 (2002), pp. 819–844.

[122] João Pedro Guerreiro Neto et al. "Turing Universality of Neural Nets (Revisited)". In: *EUROCAST '97: Proceedings of the A Selection of Papers from the 6th International Workshop on Computer Aided Systems Theory*. London, UK: Springer-Verlag, 1997, pp. 361–366.

[123] John von Neumann. *The computer and the brain*. New Haven, CT, USA: Yale University Press, 1958.

[124] Christian W. Omlin and C. Lee Giles. "Constructing Deterministic Finite-State Automata in Recurrent Neural Networks". In: *J. ACM* 43.6 (1996), pp. 937–972.

[125] Christian W. Omlin and C. Lee Giles. "Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants". In: *Neural Computation* 8.4 (1996), pp. 675–696.

[126] Toby Ord. "The many forms of hypercomputation". In: *Applied Mathematics and Computation* 178.1 (2006), pp. 143–153.

[127] Christos M. Papadimitriou. *Computational complexity*. Reading, Massachusetts: Addison-Wesley, 1994.

[128] Gheorghe Păun. "Bibliography of spiking neural P systems". In: *Natural Computing* 7.4 (2008), pp. 551–553.

[129] Gheorghe Păun. "Computing with Membranes". In: *J. Comput. Syst. Sci.* 61.1 (2000), pp. 108–143.

[130] Gheorghe Păun. *Membrane Computing. An Introduction*. Berlin: Springer-Verlag, 2002.

[131] Gheorghe Păun. "Spiking Neural P Systems: A Tutorial". In: *Bulletin of the EATCS* 91 (2007), pp. 145–159.

[132] Gheorghe Păun, Mario J. Pérez-Jiménez, and Grzegorz Rozenberg. "Spike Trains in Spiking Neural P Systems". In: *Int. J. Found. Comput. Sci.* 17.4 (2006), pp. 975–1002.

[133] Gheorghe Păun, Mario J. Pérez-Jiménez, and Arto Salomaa. "Spiking Neural P Systems: an Early Survey". In: *Int. J. Found. Comput. Sci.* 18.3 (2007), pp. 435–455.

[134] Dominique Perrin and Jean-Éric Pin. *Infinite Words – Automata, Semigroups, Logic and Games*. Vol. 141. Pure and Applied Mathematics. Elsevier, 2004.

[135] Jordan B. Pollack. "On Connectionist Models of Natural Language Processing". PhD thesis. Las Cruces, NM: Computing Reseach Laboratory, New Mexico State University, 1987.

[136] Jordan B. Pollack. “The Induction of Dynamical Recognizers”. In: *Machine Learning* 7 (1991), pp. 227–252.

[137] Yifat Prut et al. “Spatiotemporal Structure of Cortical Activity: Properties and Behavioral Relevance”. In: *Journal of Neurophysiology* 79.6 (1998), pp. 2857–2874.

[138] P. D. Roberts and C. C. Bell. “Spike timing dependent synaptic plasticity in biological systems”. In: *Biol. Cybern.* 87 (2002), pp. 392–403.

[139] Raúl Rojas. *Neural Networks - A Systematic Introduction*. Springer, 1996.

[140] Frank Rosenblatt. *The perceptron: A perceiving and recognizing automaton*. Tech. rep. 85-460-1. Ithaca, New York: Cornell Aeronautical Laboratory, 1957.

[141] Frank Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. In: 65.6 (1958), pp. 386–408.

[142] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[143] Jürgen Schmidhuber. “Deep Learning in Neural Networks: An Overview”. In: *CoRR* abs/1404.7828 (2015).

[144] Jürgen Schmidhuber. “Dynamische neuronale Netze und das fundamentale raumzeitliche Lernproblem (Dynamic neural nets and the fundamental spatio-temporal credit assignment problem)”. PhD thesis. Institut für Informatik, Technische Universität München, 1990.

[145] Victor L. Selivanov. “Fine Hierarchy of Regular Omega-Languages”. In: *Theor. Comput. Sci.* 191.1-2 (1998), pp. 37–59.

[146] H. Sebastian Seung. “Learning continuous attractors in recurrent networks”. In: *Advances in Neural Information Processing Systems*. MIT Press, 1998, pp. 654–660.

[147] C. J. Shatz. “Impulse activity and the patterning of connections during CNS development”. In: *Neuron* 5 (1990), pp. 745–756.

[148] T. Shmiel et al. “Neurons of the cerebral cortex exhibit precise interspike timing in correspondence to behavior”. In: *Proc. Natl. Acad. Sci. USA* 102.51 (2005), pp. 18655–18657.

[149] Hava T. Siegelmann. “Computation Beyond the Turing Limit”. In: *Science* 268.5210 (1995), pp. 545–548.

[150] Hava T. Siegelmann. “Neural and Super-Turing Computing”. In: *Minds Mach.* 13.1 (2003), pp. 103–114.

[151] Hava T. Siegelmann. *Neural networks and analog computation: beyond the Turing limit*. Cambridge, MA, USA: Birkhauser Boston Inc., 1999.

[152] Hava T. Siegelmann. “Recurrent Neural Networks and Finite Automata”. In: *Computational Intelligence* 12 (1996), pp. 567–574.

[153] Hava T. Siegelmann. “Stochastic Analog Networks and Computational Complexity”. In: *J. Complexity* 15.4 (1999), pp. 451–475.

[154] Hava T. Siegelmann and Eduardo D. Sontag. “Analog computation via neural networks”. In: *Theor. Comput. Sci.* 131.2 (1994), pp. 331–360.

[155] Hava T. Siegelmann and Eduardo D. Sontag. "On the computational power of neural nets". In: *J. Comput. Syst. Sci.* 50.1 (1995), pp. 132–150.

[156] Jirí Síma and Pekka Orponen. "General-Purpose Computation with Neural Networks: A Survey of Complexity Theoretic Results". In: *Neural Computation* 15.12 (2003), pp. 2727–2778.

[157] Christine A. Skarda and Walter J. Freeman. "How brains make chaos in order to make sense of the world". In: *Behavioral and Brain Sciences* 10 (02 1987), pp. 161–173.

[158] Y. Smith et al. "The thalamostriatal systems: anatomical and functional organization in normal and parkinsonian states". In: *Brain Res. Bull.* 78.2-3 (2009), pp. 60–68.

[159] Ludwig Staiger. "ω-languages". In: *Handbook of formal languages, vol. 3: beyond words*. New York, NY, USA: Springer-Verlag New York, Inc., 1997, pp. 339–387.

[160] Mike Stannett. "Computation and Hypercomputation". In: *Minds and Machines* 13.1 (2003), pp. 115–153.

[161] Mike Stannett. "The case for hypercomputation". In: *Applied Mathematics and Computation* 178.1 (2006), pp. 8–24.

[162] D. Terman et al. "Activity patterns in a model for the subthalamopallidal network of the basal ganglia". In: *J. Neurosci.* 22.7 (2002), pp. 2963–2976.

[163] *The P Systems Webpage*. http://ppage.psystems.eu/.

[164] Wolfgang Thomas. "Automata on Infinite Objects". In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Ed. by Jan van Leeuwen. Elsevier and MIT Press, 1990, pp. 133–192.

[165] S.J. Thorpe. "Spike arrival tiems: A highly efficient coding scheme for neural networks". In: *Parallel processing in neural systems*. Ed. by G. Eckmiller and G. Hauske. North-Holland: Elsevier, 1990, pp. 91–94.

[166] Ichiro Tsuda. "Chaotic itinerancy as a dynamical basis of Hermeneutics of brain and mind". In: *World Futures* 32 (1991), pp. 167–185.

[167] Ichiro Tsuda. "Hypotheses on the functional roles of chaotic transitory dynamics". In: *Chaos* 19 (2009), pp. 015113-1 –015113-10.

[168] Ichiro Tsuda. "Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems". In: *Behav. Brain Sci.* 24.5 (2001), pp. 793–847.

[169] Alan M. Turing. *Intelligent Machinery*. Technical Report. Teddington, UK: National Physical Laboratory, 1948, p. 23.

[170] Alan M. Turing. "Intelligent machinery, a heretical theory". In: *Philos. Math.* 4.3 (1996), pp. 256–260.

[171] Alan M. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem". In: *Proc. London Math. Soc.* 2.42 (1936), pp. 230–265.

[172] Alan M. Turing. *Programmers' handbook for Manchester electronic computer. Mark II*. University of Manchester. Manchester, UK, 1951.

[173] Alan Turing et al. “Can Automatic Calculating Machines Be Said To Think?” In: *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma*. Ed. by Jack B. Copeland. Oxford University Press, 2004.

[174] E. Vaadia et al. “Dynamics of neuronal interactions in monkey cortex in relation to behavioural events”. In: *Nature* 373.6514 (1995), pp. 515–518.

[175] A.E.P. Villa and J.M. Fuster. “Temporal correlates of information processing during visual short-term memory”. In: *Neuroreport* 3.1 (1992), pp. 113–116.

[176] Alessandro E. P. Villa. “Empirical Evidence about Temporal Structure in Multi-unit Recordings”. In: *Time and the brain*. Ed. by Robert Miller. Vol. 3. Conceptual Advances in Brain Research. Amsterdam, The Netherlands: Harwood Academic, 2000. Chap. 1, pp. 1–51.

[177] Alessandro E. P. Villa et al. “Chaotic dynamics in the primate motor cortex depend on motor preparation in a reaction-time task”. In: *Current Psychology of Cognition* 17 (1998), pp. 763–780.

[178] Alessandro E. P. Villa et al. “Spatiotemporal activity patterns of rat cortical neurons predict responses in a conditioned task”. In: *Proc. Natl. Acad. Sci. USA* 96.3 (1999), pp. 1106–1111.

[179] Alessandro E.P. Villa. “Spatio-temporal patterns of spike occurrences in feely-moving rats associated to perception of human vowels”. In: *Auditory cortex: Towards a synthesis of human and animal research*. Ed. by R. König et al. Mahwah, NJ: Lawrence Erlbaum, 2005, pp. 275–294.

[180] Alessandro E.P. Villa and Moshe Abeles. “Evidence for spatiotemporal firing patterns within the auditory thalamus of the cat”. In: *Brain Res* 509.2 (1990), pp. 325–327.

[181] William W. Wadge. “Reducibility and determinateness on the Baire space”. PhD thesis. University of California, Berkeley, 1983.

[182] Klaus Wagner. “On ω-regular sets”. In: *Information and Control* 43.2 (1979), pp. 123–177.

[183] Raymond L. Watrous and Gary M. Kuhn. “Induction of Finite-State Languages Using Second-Order Recurrent Networks”. In: *Neural Computation* 4.3 (1992), pp. 406–414.

[184] Michael J. Watts. “A Decade of Kasabov’s Evolving Connectionist Systems: A Review.” In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 39.3 (2009), pp. 253–269.

[185] Peter Wegner. “Interactive foundations of computing”. In: *Theor. Comput. Sci.* 192 (2 1998), pp. 315–351.

[186] Peter Wegner. “Why interaction is more powerful than algorithms”. In: *Commun. ACM* 40 (5 1997), pp. 80–91.

[187] Bernard Widrow. “The Speed of Adaption in Adaptive Control Systems”. In: *American Rocket Society (ARS) Guidance, Control and Navigation Conference Proceedings*. 1961, pp. 1933–1961.

[188] Norbert Wiener. *Cybernetics Or Control And Communication In The Animal And The Machine*. John Wiley & Sons Inc., 1948.

[189] Zheng Zeng, Rodney M. Goodman, and Padhraic Smyth. “Learning Finite State Machines With Self-Clustering Recurrent Networks”. In: *Neural Computation* 5.6 (1993), pp. 976–990.

Part II

Limit Knowledge: A Topological Approach to Epistemic Game Theory

1 INTRODUCTION

Nowadays, *game theory* has become a major field of research, mainly used in mathematical economics – towards the modelling of competing behaviors of interacting agents –, but also with considerable applications in logic, computer science, biology, political science, and psychology (see the books series AUMANN and HART (1992), AUMANN and HART (1994), AUMANN and HART (2002), YOUNG and ZAMIR (2015) for an outstanding survey of the field).

In this general context, formal *interactive epistemology* provides a general framework in which epistemic notions – such as knowledge, belief and subsequent concepts – can be modelled for situations involving multiple agents AUMANN (1999), AUMANN (1999). When employed in the specific context of game-playing agents, the discipline, referred to as *epistemic game theory*, studies the behavioral implications of such epistemic hypotheses in games, see DEKEL and SINISCALCHI (2015) and all the references there. The targeted objective of this epistemic approach to game theory consists in characterizing existing solution concepts in terms of epistemic assumptions, as well as in proposing new solution concepts by studying the consequences of refined or novel epistemic hypotheses. In fact, epistemic game theory can be regarded as complementing classical game theory. While the latter is based on the two basic primitives – game form and choice – the former adds an epistemic framework as a third elementary component such that knowledge and beliefs can be explicitly modelled in games.

In our work, we follow Aumann's set-based approach to epistemic game theory, as introduced in AUMANN (1976) and developed notably by AUMANN (1987), AUMANN (1995), AUMANN (1996), AUMANN (1998), AUMANN (1998), AUMANN (1999), AUMANN (1999), and AUMANN (2005). This approach formalizes epistemic notions via the consideration of set-theoretic tools. Events are represented as sets of possible worlds, and knowledge and belief as set-theoretic operators.

However, the standard set-based approach to interactive epistemology somehow misses a general framework providing some formal notion of closeness between events.[1] By adding a topological dimension to an epistemic structure, it is actually possible to introduce a perception of closeness of events into the reasoning

[1] For the specific purpose of dealing with counterfactuals, STALNAKER (1968) and LEWIS (1973) consider closeness between possible worlds as a primitive in the semantics of their conditional logics. The basic idea is that for every possible world and every statement, a selection function picks the closest world such that the statement holds true. In contrast to the classical models of Stalnaker and Lewis, we consider closeness between events; have closeness determined by an underlying topology; and do not restrict attention concerning closeness to counterfactuals.

of agents.[2] In such an enriched epistemic-topological framework, the reasoning of agents may thus also depend on topological instead of mere epistemic features of the underlying interactive situation.

For instance, suppose an agent is reasoning about the weather in London. Intuitively, the event *It is cloudy in London* seems to be closer to the event *It is raining in London* than the event *It is sunny in London*. Now, the agent may make identical decisions being informed only of the truth of some event within a class of *close* events. In fact, the agent might decide to stay at home not only in the case of it raining outside, but also in the case of events perceived by him to be similar, i.e. *close*, such as it being cloudy outside. In a topologically enriched epistemic framework, the notion of closeness is induced by the topology.

Note that such an epistemic-topological approach is of descriptive not normative character. Due to the heterogeneity of real-world agents, different people may have different perceptions of closeness and ways of reasoning. It therefore seems implausible to claim that there is some kind of universal topology that represents the correct perception of the event space that agents should hold. In contrast, different intuitive cognitive-topological patterns of closeness can be formalized and serve as the agents' perceptions of the event space. Such a descriptive use of topologies is in line with RUBINSTEIN (1989)'s view that topology can be used as a substantial tool to formalize natural intuitions about closeness.

In line with these consideration, BACH and CABESSA (2011), BACH and CABESSA (2012), BACH and CABESSA (2009), BACH and CABESSA (2016) introduced a topological approach to interactive epistemology and epistemic game theory. They consider Aumann structures equipped with topologies on the event space in order to capture some notion of closeness between events. On that basis, they introduce the epistemic-topological operator *limit knowledge*, defined as the topological limit of all higher-order mutual knowledge claims, and accordingly, linked to both the epistemic as well as the topological features of the underlying semantics.

They further study some game-theoretic consequences of limit knowledge, and show that this epistemic-topological operator is capable of relevant characterizations of solution concepts in games BACH and CABESSA (2012), BACH and CABESSA (2009). Besides, they revisit AUMANN (1976)'s "no-agreeing to disagree theorem" in their enriched epistemic-topological context, and prove that the impossibility to agree to disagree does no longer hold when the epistemic hypothesis of common knowledge of the posteriors is replaced by that of limit knowledge of the posteriors BACH and CABESSA (2011), BACH and CABESSA (2016). These considerations argue in favor of a general topological approach to set-based interactive epistemology.

The following chapters present this topological approach to interactive epistemology and epistemic game theory. Chapter 2 recalls the basics of the set-based approach to interactive epistemology. Chapter 3 introduces the epistemic-topological operator *limit knowledge* and emphasizes its difference with *common knowledge*. Chapter 4 studies the behavioral implications of limit knowledge of rationality in game theory. Chapter 5 revisit AUMANN (1976)'s "no agreeing to disagree" theo-

[2] We recall that topological spaces are generalizations of metric spaces. While closeness between elements is explicitly measured via the respective distance function in metric spaces, it is only implicitly determined the coleection of open sets in topological spaces.

rem from the limit knwoledge perspective. Finally, Chapter 6 provides some concluding remarks.

2 SET-BASED APPROACH TO INTERACTIVE EPISTEMOLOGY

2.1 AUMANN STRUCTURES

The so-called set-based approach to interactive epistemology has been introduced and notably developed by AUMANN (1976), AUMANN (1987), AUMANN (1995), AUMANN (1999), AUMANN (1999) and AUMANN (2005). This approach formalizes epistemic notions via the consideration of set-theoretic tools. Events are represented as sets of possible worlds, and knowledge and belief as set-theoretic operators.

The basic ingredient of this set-based approach is the concept of an Aumann structure.

Definition 1. An *Aumann structure* consists of a tuple $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in I}, p)$, where:

- Ω is a countable set of possible worlds;
- I is a finite set of agents;
- for each agent $i \in I$, $\mathcal{I}_i$ is a possibility partition of Ω;
- $p : \Omega \to [0,1]$ is a common prior belief function satisfying $\sum_{\omega \in \Omega} p(\omega) = 1$.

The elements of Ω are called *possible worlds*, or *states*, and provide a complete descriptions of the way the world might be. The Aumann structure is called finite if Ω is finite and infinite otherwise. The possibility partition $\mathcal{I}_i$ of Ω, for each agent $i \in I$, represents the *agent's information*. The cell of $\mathcal{I}_i$ containing the world ω is denoted by $\mathcal{I}_i(\omega)$ and consists of all worlds considered possible by i at world ω. In other words, agent i cannot distinguish between any two worlds ω and ω' that are in the same cell of his partition $\mathcal{I}_i$. Two such worlds are called *indistinguishable for agent i*. Equivalently, if $\mathcal{I}_i(\omega) \neq \mathcal{I}_i(\omega')$, then ω and ω' are said to be *distinguishable for agent i*. We then call two worlds ω and ω' *distinguishable* if they are distinguishable for all agents $i \in I$. The *common prior belief function* $p : \Omega \to [0,1]$ represents probability that each possible world $\omega \in \Omega$ corresponds to the real (or actual) world.

In this context, an *event* is defined as a set $E \subseteq \Omega$ of possible worlds. For instance, the event that it is raining in London would be represented by the set of all worlds in which it does in fact rain in London. The common prior belief function p is naturally extended to a common prior belief measure on the event space

$p : \mathcal{P}(\Omega) \to [0,1]$ by setting $p(E) = \sum_{\omega \in E} p(\omega)$. This probability measure represents the probability that each possible event $E \subseteq \Omega$ does actually occur in the real world.

Besides, it is supposed that each information set of each agent has non-zero prior probability, i.e., $p(\mathcal{I}_i(\omega)) > 0$, for all $i \in I$ and $\omega \in \Omega$. Such a hypothesis seems plausible since it ensures that no information is excluded a priori. It is also assumed that all agents are Bayesian, and hence, update the common prior belief given their private information according to Bayes' rule. More precisely, given some event E and some world ω, the posterior belief of agent i in E at ω is given by

$$p(E \mid \mathcal{I}_i(\omega)) = \frac{p(E \cap \mathcal{I}_i(\omega))}{p(\mathcal{I}_i(\omega))}.$$

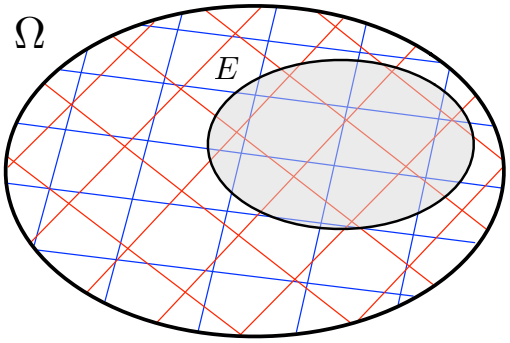


Figure 1 – Illustration of an Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in \{1,2\}}, p)$ involving two agents. The blue and red partitions represent the information of agent 1 and 2, respectively. The set $E \subset \Omega$ represents a possible event.

Example 2. A simple Aumann structure is illustrated in Figure 3 (at the end of the chapter).

2.2 Aumann Structures in Game Theory

Aumann structures can also be used as epistemic models for games. In this context, the Aumann structure needs an additional component that connects the interactive epistemology to the game. Before stating the formal definition, the general notion of a game is recalled.

Definition 3. A *game* consists of a tuple $\Gamma = (I, (S_i)_{i \in I}, (u_i)_{i \in I})$, where:

- I is a set of *players*;
- S_i is the *strategy set* of player i, for each $i \in I$, ;
- $u_i : \prod_{i \in I} S_i \to \mathbb{R}$ is the *utility function* of player i, for each $i \in I$;

The set of players I is countable and each strategy sets S_i is possibly uncountable. The utility function $u_i : \prod_{i \in I} S_i \to \mathbb{R}$ assigns to each strategy profile $(s_i)_{i \in I} \in \prod_{i \in I} S_i$ of the players a certain level of satisfaction $u_i((s_i)_{i \in I}) \in \mathbb{R}$ of player i.

Aumann structures, when being employed as epistemic models for games, specify an additional choice function for each player i that connects the interactive epistemology to the game.

Definition 4. Let $\Gamma = (I, (S_i)_{i\in I}, (u_i)_{i\in I})$ be some game. An epistemic model of Γ is an augmented Aumann structure $\mathcal{A}^\Gamma = (\Omega, (\mathcal{I}_i)_{i\in I}, (\sigma_i)_{i\in I})$ where:

- Ω is a set of possible worlds;
- I is a set of players;
- $\mathcal{I}_i$ is a possibility partition of Ω, for each player $i \in I$;
- $\sigma_i : \Omega \to S_i$ is a choice function, for each player $i \in I$.

As for classical Aumann structures, Ω represents the set of possible worlds and $(\mathcal{I}_i)_{i\in I}$ the information partitions of the players. The additional *choice function* $\sigma_i : \Omega \to S_i$ assigns to each possible world $\omega \in \Omega$ a definite strategy $\sigma_i(\omega)$ that player i will follow if the actual world would correspond to ω. The corresponding *choice function profile* $\sigma : \Omega \to \prod_{i\in I} S_i$, mapping each possible world to its corresponding strategy profile, is naturally defined by $\sigma(\omega) = (\sigma_i(\omega))_{i\in I}$.

In this context, it is standard to assume that each player knows his own strategy choice. This so-called measurability assumption seems natural in the context of game theory, where agents make their choices deliberately and consciously. AUMANN and BRANDENBURGER (1995) even denote it as tautologous by pointing out that knowing one's own choice is implicit in consciously making a choice. Formally, the measurability assumption requires each player's choice function σ_i to be measurable with respect to $\mathcal{I}_i$, i.e. if two worlds ω and ω' are in the same cell of player i's information partition, then $\sigma_i(\omega) = \sigma_i(\omega')$.

Example 5. A simple game and epistemic model of it is illustrated in Example 10 (at the end of the chapter).

2.3 KNOWLEDGE

In Aumann structures, knowledge is formalized as a set-theoretic operator K_i which, to any event E, associates a corresponding event "agent i knows E".

Definition 6. Given some event E, the event that *agent i knows E* is defined as

$$K_i(E) := \{\omega \in \Omega : \mathcal{I}_i(\omega) \subseteq E\}.$$

Equivalently, $K_i(E)$ corresponds to the union of all partition cells of $\mathcal{I}_i$ that are included in E, as illustrated in Figure 2. Intuitively, agent i knows some event E if, in all worlds he considers possible, E holds. Agent i is said to know E at world ω if $\omega \in K_i(E)$. Naturally, the *mutual knowledge of E* among the set I of agents is defined as the event $K(E) := \bigcap_{i\in I} K_i(E)$. Iterated mutual knowledge can then be formalized inductively. More precisely, letting $K^0(E) := E$, *m-order mutual knowledge of the event E* among the set I of agents is defined by $K^m(E) := K(K^{m-1}(E))$ for all $m > 0$. Accordingly, mutual knowledge can also be denoted as 1-order mutual knowledge. Knowledge and mutual knowledge are illustrated in Figure 2.

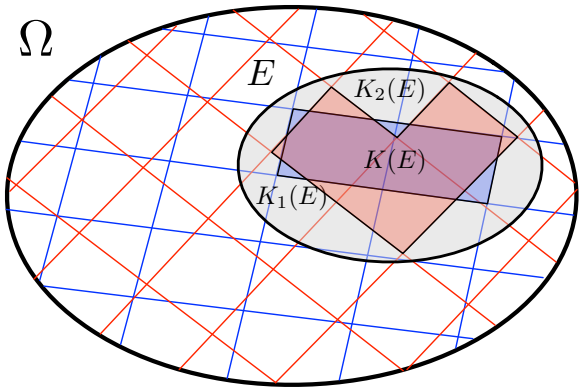


Figure 2 – Illustration of the concepts of knowledge and mutual knowledge in an Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in\{1,2\}}, p)$ involving two agents. The event $K_1(E)$ and $K_2(E)$ and the unions of the blue and red partition cells that are included in E, respectively. The event $K(E)$ is the intersection of $K_1(E)$ and $K_2(E)$.

The belief operator is naturally defined as the dual of the knowledge one. Formally, *agent i believes E* if and only if she doesn't know the negation of E, i.e.,

$$B_i(E) := (K_i(E^c))^c,$$

where X^c denotes the complement of X.

One can show that the knowledge operator K_i satisfies the following properties:

$K_i(E) \subseteq E$ (Factiveness or Truth Axiom)

$K_i(E) \subseteq K_i(K_i(E))$ (Positive Introspection)

$(K_i(E))^c \subseteq K_i((K_i(E))^c)$ (Negative Introspection)

If $E \subseteq F$ then $K_i(E) \subseteq K_i(F)$ (Monotonicity)

If $(E_n)_{n\in N}$ is a decreasing sequence of events, then

$$K_i(\bigcap_{n\in N} E_n) = \bigcap_{n\in N} K_i(E_n) \quad \text{(P1)}$$

If $n \geq m$, then $K_i^n(E) \subseteq K_i^m(E)$ (P2)

The "factiveness" or "truth axiom" says that if i knows E, then E necessarily holds, i.e., it is not possible to know false things. In fact, the notable contrast between knowledge and belief resides in the very fact that false claims cannot be known, yet can be believed. The "positive introspection" states that if i knows E, then she knows that she knows E. The "negative introspection" states that if i doesn't know E, then she knows that she doesn't know E. The "monotonicity" means that if E implies F and i knows E, then i also knows F, which means that agents never make make wrong inferences. Property $P1$ states that, when decreasing sequences are considered, the knowledge operator commutes with the infinite intersection. Property $P2$ states that the sequence of iterated mutual knowledge

claims is decreasing, and therefore, satisfies Property $P1$. It generalizes the characteristic property of knowledge – the truth axiom – to arbitrary higher-order mutual knowledge.

Finally, from Property $P2$, one can infer that any sequence of iterated mutual knowledge $(K^m(E))_{m>0}$ is either *strictly shrinking*, i.e., $K^{m+1}(E) \subsetneq K^m(E)$ for all $m \geq 0$, or *eventually constant*, i.e., there exists some index p such that $K^m(E) = K^p(E)$ for all $m \geq p$. Indeed, Property $P2$ ensures that the sequence is shrinking, i.e., $K^{m+1}(E) \subseteq K^m(E)$, for all $m \geq 0$. Now, suppose that the sequence is not strictly shrinking, i.e., there exists $p \geq 0$ such that $K^{p+1}(E) = K^p(E)$. It follows by induction on n that $K^{p+n}(E) = K^p(E)$, for all $n > 0$. Hence, the sequence is eventually constant. The case of sequences of iterated mutual knowledge being strictly shrinking will be of specific importance in the sequel.

Example 7. An simple case of the concepts of knowledge and mutual knowledge is illustrated in Example 10 (at the end of the chapter).

2.4 Common Knowledge

The notion of "common knowledge" is one of the central epistemic concepts in game theory. It is used in basic background assumptions, such as common knowledge of the game form, as well as in epistemic hypotheses, such as common knowledge of rationality, which in turn can be applied to epistemic foundations of solution concepts.

The notion has originally been introduced by Lewis (1969) as a prerequisite for a rule to become a convention. Intuitively, some event is said to be common knowledge among a set of agents, if everyone knows the event, everyone knows that everyone knows the event, everyone knows that everyone knows that everyone knows the event, etc., ad infinitum. In his seminal paper, Aumann (1976) writes:

> Call two people 1 and 2. When we say that an event is "common knowledge", we mean more than just both 1 and 2 know it; we require also that 1 knows that 2 knows it, 2 knows that 1 knows it, 1 knows that 2 knows that 1 knows it, and so on. For example, if 1 and 2 are both present when the event happens and see each other there, then the event becomes common knowledge.

It has therefore become standard to define common knowledge of an event as the infinite intersection, or conjunction, of all iterated mutual knowledge claims of that event. These consideration lead to the so-called iterative definition of common knowlegde.

Definition 8. Let $\mathcal{A}$ be an Aumann structure and E be some event. The event that E is *common knowledge among agents* is defined as

$$CK^{iter}(E) := \bigcap_{m>0} K^m(E).$$

Accordingly, we say that E *is common knowledge at world* ω among the set I of agents iff $\omega \in CK^{iter}(E)$.

An alternative formalization of common knowledge is proposed by AUMANN (1976) in terms of the meet of the agents' possibility partitions.[1] Formally, common knowledge of some event E can be stated as

$$CK^{meet}(E) := \{\omega \in \Omega : (\bigwedge_{i \in I} \mathcal{I}_i)(\omega) \subseteq E\}.$$

Alternatively put, $CK^{meet}(E)$ is the union of all the meet's cells that are contained in E. Hence, an event E will be called *common knowledge at world* ω among the set I of agents iff $(\bigwedge_{i \in I} \mathcal{I}_i)(\omega) \subseteq E$, or in words, iff E includes the cell of the meet $\bigwedge_{i \in I} \mathcal{I}_i$ that contains ω.

A third fixed-point characterization of common knowledge has also been proposed in the literature, for instance by MONDERER and SAMET (1989). This fixed point view is in fact implicit in AUMANN (1976)'s meet definition of common knowledge. Here, common knowledge of some some event is defined as the claim that everyone knows both the event and common knowledge of the event. Common knowledge of E is therefore a solution of the set-theoretic equation:

$$X = K(E \cap X).$$

A non self-referential definition of this concept can actually be given. It is based on the notion of an evident knowledge event. We say that E is an *evident knowledge* event if whenever it occurs, all agents know it, i.e., if $E \subseteq K_i(E)$, for each agent $i \in I$, or equivalently if $E \subseteq K(E)$. Note that the factiveness of knowledge (cf. Section 2.3) ensures that $K(E) \subseteq E$, and hence, evident knowledge events actually satisfy $E = K(E)$. These events are consequently fixed-points of the knowledge operator, as well as of any iteration if the knowledge operator, i.e., $E = K^m(E)$, for all $m \geq 0$.[2] The typical evident knowledge events are public announcements: as soon as they happen, everyone necessarily knows them. The fixed-point definition of common knowledge of some event E is then stated as follows:

$$CK^{fp}(E) := \{\omega \in \Omega : \text{there exists some evident knowledge event } E' \text{ satisfying } \omega \in E' \subseteq K(E)\}$$

In words, $CK^{fp}(E)$ is the union of all evident knowledge events that are contained in $K(E)$. Accordingly, the event E is said to be *common knowledge at world* ω among the set I of agents iff there exists an evident knowledge event E' such that $\omega \in E' \subseteq K(E)$.

The natural question then arises whether these three definitions are equivalent. BARWISE (1988) provided a special situation-theoretic model in which the standard and fixed point views of common knowledge do not coincide. Moreover, SARENAC (2008) also showed the non-equivalence of the two notions within a framework of epistemic logic with topological semantics. However, in Aumann structures, the

[1] Given two partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ of a set S, partition $\mathcal{P}_1$ is called *finer* than partition $\mathcal{P}_2$ or $\mathcal{P}_2$ *coarser* than $\mathcal{P}_1$, if each cell of $\mathcal{P}_1$ is a subset of some cell of $\mathcal{P}_2$. Given n partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ of S, the finest partition that is coarser than $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ is called the *meet* of $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ and is denoted by $\bigwedge_{j=1}^{n} \mathcal{P}_j$. Moreover, given $x \in S$, the cell of the meet $\bigwedge_{j=1}^{n} \mathcal{P}_j$ containing x is denoted by $(\bigwedge_{j=1}^{n} \mathcal{P}_j)(x)$.

[2] The property is easily proven by induction on m.

three notions do coincide. This equivalence (between the iterative and meet definitions) was already informally stated by AUMANN (1976), and is properly shown by the following Lemma. Consequently, for any event E, the three identical events $CK^{iter}(E)$, $CK^{meet}(E)$ and $CK^{fp}(E)$ will simply be referred to as $CK(E)$ in the sequel.

Lemma 9. *Let $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$ be an Aumann structure and $E \subseteq \Omega$ be an event. Then, $CK^{iter}(E) = CK^{meet}(E) = CK^{fp}(E)$.*

Proof. Firstly, we show that $CK^{meet}(E) \subseteq CK^{iter}(E)$. By definition, $CK^{meet}(E)$ is the union of the cells of the meet $\bigwedge_{i\in I}\mathcal{I}_i$ that are included in E. Since the meet $\bigwedge_{i\in I}\mathcal{I}_i$ is coarser than each agent's possibility partition, the event $CK^{meet}(E)$ can thus be written as a union of information cells for each agent $i \in I$, i.e. for all $i \in I$, there exists a set $F_i \subseteq \Omega$ such that $CK^{meet}(E) = \bigcup_{\omega'\in F_i}\mathcal{I}_i(\omega')$. It follows that $K_i(CK^{meet}(E)) = K_i(\bigcup_{\omega'\in F_i}\mathcal{I}_i(\omega')) = \bigcup_{\omega'\in F_i}\mathcal{I}_i(\omega') = CK^{meet}(E)$, for all $i \in I$. Hence, $K(CK^{meet}(E)) = \bigcap_{i\in I}K_i(CK^{meet}(E)) = \bigcap_{i\in I}CK^{meet}(E) = CK^{meet}(E)$. It follows by induction that $K^m(CK^{meet}(E)) = CK^{meet}(E)$, for all $m > 0$. Hence, $CK^{iter}(CK^{meet}(E)) = \bigcap_{m>0}K^m(CK^{meet}(E)) = \bigcap_{m>0}CK^{meet}(E) = CK^{meet}(E)$, which shows that $CK^{meet}(E)$ is a fixed point of the CK^{iter} operator. Finally, since the operator CK^{iter} is monotone with respect to set inclusion and since $CK^{meet}(E) \subseteq E$, it follows that $CK^{meet}(E) = CK^{iter}(CK^{meet}(E)) \subseteq CK^{iter}(E)$.

Secondly, we show that $CK^{iter}(E) \subseteq CK^{meet}(E)$. As a preliminary claim, we prove by induction on $n \in \mathbb{N}$ that, for any $\omega, \omega^* \in \Omega$, if both $\omega^* \notin E$ and there exists a sequence of $n+1$ possibility cells $(\mathcal{I}_k)_{k=0}^n$ such that $\omega \in \mathcal{I}_0$, $\omega^* \in \mathcal{I}_n$, and $\mathcal{I}_k \cap \mathcal{I}_{k+1} \neq \emptyset$ for all $0 \leq k < n$, then $\omega \notin K^{n+1}(E)$. First of all, for the case $n = 0$, let $\omega, \omega^* \in \Omega$ and $\mathcal{I}_0$ be a possibility cell such that $\omega, \omega^* \in \mathcal{I}_0$, and $\omega^* \notin E$. Since $\omega \in \mathcal{I}_0$, the cell $\mathcal{I}_0$ can be written as $\mathcal{I}_{i_0}(\omega)$, where $i_0 \in I$ denotes the agent to whom the cell $\mathcal{I}_0$ belongs. Also, $\omega^* \in \mathcal{I}_0 = \mathcal{I}_{i_0}(\omega)$ and $\omega^* \notin E$ imply that $\mathcal{I}_{i_0}(\omega) \not\subseteq E$. It follows that $\omega \notin K_{i_0}(E)$ and hence $\omega \notin \bigcap_{i\in I}K_i(E) = K(E) = K^1(E)$. Now, assume that the claim holds true for some $n = m$. Let $\omega, \omega^* \in \Omega$ and let $(\mathcal{I}_k)_{k=0}^{m+1}$ be a sequence of $m+2$ possibility cells such that $\omega^* \notin E$, $\omega \in \mathcal{I}_0$, $\omega^* \in \mathcal{I}_{m+1}$, and $\mathcal{I}_k \cap \mathcal{I}_{k+1} \neq \emptyset$ for all $0 \leq k < m+1$. Since $\mathcal{I}_0 \cap \mathcal{I}_1 \neq \emptyset$, there exists some world $\omega' \in \mathcal{I}_0 \cap \mathcal{I}_1$. Now consider the sequence of $m+1$ possibility cells $(\mathcal{J}_k)_{k=0}^m$ defined by $\mathcal{J}_k = \mathcal{I}_{k+1}$ for all $0 \leq k \leq m$. This sequence satisfies $\omega' \in \mathcal{J}_0$, $\omega^* \in \mathcal{J}_m$, and $\mathcal{J}_k \cap \mathcal{J}_{k+1} \neq \emptyset$ for all $0 \leq k < m$. By the induction hypothesis, it follows that $\omega' \notin K^{m+1}(E)$. Moreover, since $\omega' \in \mathcal{I}_0$, the cell $\mathcal{I}_0$ can be written as $\mathcal{I}_{i_0}(\omega')$, where $i_0 \in I$ denotes the agent to whom the cell $\mathcal{I}_0$ belongs. Hence, $\omega' \in \mathcal{I}_0 = \mathcal{I}_{i_0}(\omega)$ and $\omega' \notin K^{m+1}(E)$ imply that $\mathcal{I}_{i_0}(\omega) \not\subseteq K^{m+1}(E)$. It follows that $\omega \notin K_{i_0}(K^{m+1}(E))$ and hence $\omega \notin \bigcap_{i\in I}K_i(K^{m+1}(E)) = K(K^{m+1}(E)) = K^{m+2}(E)$, which completes the proof of the preliminary claim. Now, let $\omega \notin CK^{meet}(E)$. By the definition of $CK^{meet}(E)$, it holds that $(\bigwedge_{i\in I}\mathcal{I}_i)(\omega) \not\subseteq E$. Since the meet $\bigwedge_{i\in I}\mathcal{I}_i$ is the finest partition coarser than all agents' possibility partitions, the cell $(\bigwedge_{i\in I}\mathcal{I}_i)(\omega)$ can be written as a union of consecutively intersecting distinct cells, i.e. there exists an index set $K \subseteq \mathbb{N}$ and a sequence of possibility cells $(\mathcal{I}_k)_{k\in K}$ such that $(\bigwedge_{i\in I}\mathcal{I}_i)(\omega) = \bigcup_{k\in K}\mathcal{I}_k$, as well as $\omega \in \mathcal{I}_0$ and $\mathcal{I}_{k-1} \cap \mathcal{I}_k \neq \emptyset$, for all $k \in K \setminus \{0\}$. Note that any two consecutive terms of the sequence $(\mathcal{I}_k)_{k\in K}$ actually belong to distinct agents, since $\mathcal{I}_{k-1} \cap \mathcal{I}_k \neq \emptyset$ ensures that $\mathcal{I}_{k-1}$ and $\mathcal{I}_k$ are not in a same agent's possibility partition. As $(\bigwedge_{i\in I}\mathcal{I}_i)(\omega) \not\subseteq E$, there exists a world ω^* such that $\omega^* \in (\bigwedge_{i\in I}\mathcal{I}_i)(\omega)$ and $\omega^* \notin E$. Moreover, since $\omega^* \in (\bigwedge_{i\in I}\mathcal{I}_i)(\omega) = \bigcup_{k\in K}\mathcal{I}_k$,

there exists an index $l \in K$ such that $\omega^* \in \mathcal{I}_l$. Therefore, $\omega^* \notin E$, and the sequence of $l+1$ possibility cells $(\mathcal{I}_k)_{k=0}^{l}$ satisfies $\omega \in \mathcal{I}_0$, $\omega^* \in \mathcal{I}_l$, as well as $\mathcal{I}_k \cap \mathcal{I}_{k+1} \neq \emptyset$, for all $0 \leq k < l$. By the preliminary claim, it thus follows that $\omega \notin K^{l+1}(E)$. Hence, $\omega \notin \bigcap_{m>0} K^m(E) = CK^{iter}(E)$.

We finally show that $CK^{iter}(E) = CK^{fp}(E)$. Let $\omega \in CK^{iter}(E) = \bigcap_{m>0} K^m(E)$. By taking $E' = CK^{iter}(E)$, one has, by Property $P1$ of knowledge, that E' is an evident event. Moreover, E' satisfies $\omega \in E' \subseteq K(E)$. Hence, $w \in CK^{fp}(E)$. Conversely, let $\omega \in CK^{fp}(E)$. Then, there exists an evident event E' such that $\omega \in E' \subseteq K(E)$. Since E' is evident knowledge and by the monotonicity of the knowledge operator, one has $E' = K(E') \subseteq K^2(E)$. By induction on m, it follows that $E' \subseteq K^m(E)$, for all $m > 0$, and thus $E' \subseteq \bigcap_{m>0} K^m(E) = CK^{iter}(E)$. Since $\omega \in E'$, one has $\omega \in CK^{iter}(E)$. □

We finally provide a simple example which illustrates all the previous notions.

Example 10. Consider the case of a two player prisoner's dilemma. Each player might either cooperate or defect, and the playoffs corresponding to each strategy profile of the players are given in Table 1 below. A possible epistemic model of this simple game is given by the Aumann structure of Figure 3. In terms of the strategy profiles of the players, there are only four possible worlds represented by each cell of the payoff matrix. In this case, the choice functions σ_i are naturally given by $\sigma_1((x;y)) = x$ and $\sigma_2((x;y)) = y$, for all $x, y \in \{C, D\}$, i.e., at world $(x;y)$ Player 1 and 2 play strategies x and y, respectively.

	Cooperate (C)	Defect (D)
Cooperate (C)	$(1;1)$	$(3;0)$
Defect (D)	$(0;3)$	$(2;2)$

Table 1 – Payoff matrix of the game of the prisoner's dilemma.

In this specific case, the informations of Player 1 and 2 are not symmetric. Indeed, Player 1 can distinguish between her own strategies, but cannot distinguish between those of her opponent: the worlds $(C;C)$ and $(C;D)$ belong to a same possibility cell of hers, and the worlds $(D;C)$ and $(D;D)$ also. By contrast, Player 2 can distinguish between his own strategies, and, only in the specific case where he defects, he can also distinguish between those of his opponent: the worlds $(C;C)$ and $(D;C)$ belong to a same possibility cell, but the worlds $(C;D)$ and $(D;D)$ are isolated into two possibility cells.

The event E that "Player 2 defect" is composed of the two possible worlds where player 2 does defect, i.e., $E = \{(C;D), (D;D)\}$. At worlds $(C;D)$ and $(D;D)$, Player 2 knows that she defects, since $K_2(E) = \{(C;D), (D;D)\}$. However, player 1 never knows that player 2 defects, since $K_2(E) = \emptyset$. Hence, the players never mutual know that Player 2 defects, since $K(E) = K_1(E) \cap K_2(E) = \emptyset$, and a fortiori, it is never common knowledge that Player 2 defects, since $CK(E) = \emptyset$.

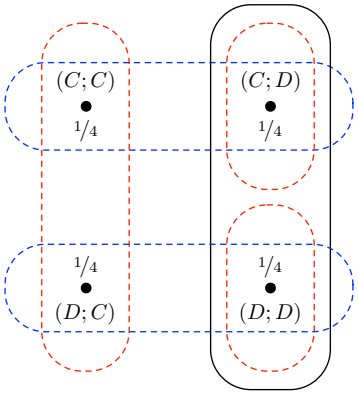


Figure 3 – An Aumann structure providing an epistemic model for the game of the prisoner's dilemma given by Table 1. The choice functions σ_i are naturally given by $\sigma_1((x;y)) = x$ and $\sigma_2((x;y)) = y$, for all $x, y \in \{C, D\}$, i.e., at world $(x;y)$ Player 1 and 2 play strategies x and y, respectively. The blue and red partitions represent the informations of Player 1 and 2, respectively. The prior beliefs are given by the fractions associated to each possible world. Here, we assume that the players chose their strategies randomly, and therefore, each of the four worlds are equiprobable. The event that "Player 2 defect", composed of the two possible worlds where player 2 does defect, is illustrated by the black set.

3 LIMIT KNOWLEDGE

3.1 INTRODUCTION

Apart from comparing distinct conceptions of common knowledge, a further related question concerns the relationship between the standard definition of common knowledge and the infinite sequence of iterated mutual knowledge underlying it. In fact, LIPMAN (1994) showed that common knowledge of the particular event rationality is not equivalent to the limit of iterated mutual knowledge for some specific notion of limit.

Here, we also study the relationship between common knowledge and the limit of the sequence of iterated mutual knowledge, but from a topological point of view.[1] More precisely, we introduce the epistemic-topological operator "limit knowledge", defined as the topological limit of the the sequence of iterated mutual knowledge claims. We further show that "common knowledge" and "limit knowledge" do genuinely differ.

3.2 LIMIT KNOWLEDGE

The epistemic-topological operator "limit knowledge" is defined as the topological limit of the sequence of higher-order mutual knowledge claims, according to some topology on the event space.

Definition 11. Let $(\Omega, (\mathcal{I}_i)_{i\in I}, p)$ be an Aumann structure, $\mathcal{T}$ a topology on $\mathcal{P}(\Omega)$, and E an event. If the limit point of the sequence $(K^m(E))_{m>0}$ 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.

Hence, limit knowledge of an event E is constituted by – whenever unique – the limit point of the sequence of iterated mutual knowledge, and thus linked to both epistemic as well as topological aspects of the event space.

[1] For sake of self-containedness, some basic notions from topology are recalled. Given some set X, a *topology* $\mathcal{T}$ on X consists of a family of subsets of X, i.e. $X \subseteq \mathcal{P}(X)$, such that the empty set and X belong to $\mathcal{T}$, and the family $\mathcal{T}$ is closed under finite intersection as well as under arbitrary union. If $\mathcal{T}$ is a topology on X, the pair $(X, \mathcal{T})$ is called *topological space*, where the elements of $\mathcal{T}$ are called *open sets*. For any $p \in X$, an *open neighbourhood* of p is an open set containing p. Moreover, given some subset $S \subseteq X$, an element $p \in X$ is a *limit point* of S if every open neighbourhood of p contains some element of S different from p. Given some sequence $s = (x_i)_{i\geq 0}$ of elements of X, if the limit point of s is unique, it is denoted by $\lim_{i\to\infty} x_i$.

Limit knowledge of an event can be understood as the event which is approached by the sequence of iterated mutual knowledge, according to the notion of closeness between events provided by a given topology on the event space.[2] Thus, the higher the iterated mutual knowledge, the closer this epistemic event is to limit knowledge.

It is possible to link limit knowledge to reasoning patterns of agents based on closeness of events. In fact, agents satisfying limit knowledge of some event are in a situation infinitesimally close to having arbitrarily-high iterated mutual knowledge of this event, and the agents' reasoning may be influenced accordingly. Note that a reasoning pattern associated with limit knowledge depends on the particular topology on the event space, which fixes the closeness relation between events.

Although being more and more proximal to iterated mutual knowledge the higher the iteration, it is possible – depending on the topology – that limit knowledge is not included in all higher-order mutual knowledge or even in the underlying event itself (see Example 15). Therefore, limit knowledge does not a priori inherit the purely epistemic properties of higher-order mutual knowledge or even knowledge. Actually, agents entertaining limit knowledge of some event might notably be in situations in which the event does not hold, while at the same time being arbitrarily close to the highest iterated mutual knowledge of the event.

Generalizations of the concept of limit knowledge could be conceived in order to overcome the undefinability of this operator in cases of non unique limit points. For instance, *multiple-limit knowledge of E* could be defined as the union of all limit points of $(K^m(E))_{m>0}$.

3.3 Limit knowledge vs Common Knowledge

The two epistemic and topological-epistemic operators of "common knowledge" and "limit knowledge" are both grounded on the sequence of iterated mutual knowledge claims. Hence, a natural question to be addressed is to clarify the relationship between the two operators. We show that these two concepts are closely related for finite, yet do substantially differ for infinite Aumann structures.

We first focus on finite Aumann structures. In this case, for any topology on the event space, common knowledge of an event E is always a limit point of the sequence of higher-order mutual knowledge of E, as established by the following result.

Proposition 12. *Let $\mathcal{A}$ be a finite Aumann structure, $\mathcal{T}$ be a topology on $\mathcal{P}(\Omega)$, and E be some event. Then, $CK(E)$ is a limit point of $(K^m(E))_{m>0}$. In particular, if the limit point of $(K^m(E))_{m>0}$ is unique, then $CK(E) = LK(E)$.*

Proof. Note that since Ω is finite, its power set $\mathcal{P}(\Omega)$ is also finite. Moreover, Prop-

[2] In our epistemic-topological framework, there exist various ways in which a notion of closeness between events can be defined based on the underlying topology, and in particular, on the open neighbourhoods. For instance, two events could be said to be close, if they violate the so-called Hausdorff-condition, i.e., if there exist no disjoint neighbourhoods of these two events. Also, degrees of closeness could possibly be defined in the sense that the more neighbourhoods contain two events, the closer the respective events are to each other. In case of the considered topology on the event space to be metrizable, there exists a distance function which could explicitly measure the closeness between events.

erty $P2$ of knowledge ensures that $K^{m+1}(E) \subseteq K^m(E)$ for all $m > 0$. Thus, by finiteness of $\mathcal{P}(\Omega)$, the sequence $(K^m(E))_{m>0}$ is eventually constant, i.e. there exists some index p such that $K^m(E) = K^p(E)$ for all $m \geq p$. Hence $CK(E) = \bigcap_{m>0} K^m(E) = \bigcap_{m \geq p} K^m(E) = K^p(E)$. Moreover, for any $\mathcal{T}$-open neighbourhood N of $CK(E)$, it holds that $K^m(E) = K^p(E) = CK(E) \in N$, for all $m \geq p$. Therefore, $CK(E)$ is a limit point of the sequence $(K^m(E))_{m>0}$. If the limit point of $(K^m(E))_{m>0}$ is unique, then it is $LK(E)$ by definition, and thus $CK(E) = LK(E)$. □

Note that the sequence $(K^m(E))_{m>0}$ may converge to multiple limit points, and $CK(E)$ is always one of them. However, if $\mathcal{P}(\Omega)$ is equipped with a Hausdorff topology, then the limit of $(K^m(E))_{m>0}$ is necessarily unique, and thus $CK(E) = LK(E)$. Since the discrete topology is the only Hausdorff topology on finite spaces, one has $CK(E) = LK(E)$ in this case.

Now, infinite Aumann structures are considered. The following result shows that if the sequence of iterated mutual knowledge is eventually constant[3], then common knowledge is always one of its limit points.

Proposition 13. *Let $\mathcal{A}$ be an infinite Aumann structure, $\mathcal{T}$ be a topology on $\mathcal{P}(\Omega)$, and E be some event. If $(K^m(E))_{m>0}$ is eventually constant, then $CK(E)$ is a limit point of $(K^m(E))_{m>0}$. In particular, if the limit point of $(K^m(E))_{m>0}$ is unique, then $CK(E) = LK(E)$. If the limit point of $(K^m(E))_{m>0}$ is unique, then it is $LK(E)$ by definition, and thus $CK(E) = LK(E)$.*

Proof. Suppose that the sequence $(K^m(E))_{m>0}$ is eventually constant from index p onwards. By Property $P2$ of knowledge, it follows that $CK(E) = \bigcap_{m>0} K^m(E) = \bigcap_{m>p} K^m(E) = K^p(E)$. Now let N be a $\mathcal{T}$-open neighborhood of $CK(E)$. Since $K^m(E) = K^p(E) = CK(E)$, for all $m \geq p$, it follows that $K^m(E) \in N$, for all $m \geq p$. Therefore, $CK(E)$ is a limit point of the sequence $(K^m(E))_{m>0}$. □

It follows that common knowledge and limit knowledge can only possibly be distinct in the case of the sequence of iterated mutual knowledge not being eventually constant. Since the latter sequence is either eventually constant or strictly shrinking, potential differences of the two concepts necessarily require the *strictly shrinking condition* to be met.

In case of the sequence of iterated mutual knowledge being strictly shrinking, common knowledge and limit knowledge may indeed differ.

Proposition 14. *There exist an infinite Aumann structure $\mathcal{A}$, a topology on the event space $\mathcal{P}(\Omega)$, and some event E, such that $CK(E) \neq LK(E)$.*

Proof. Consider the infinite Aumann structure $\mathcal{A} = (\mathbb{N}, (\mathcal{I}_i)_{i \in \{Alice, Bob\}})$ given by

$$\begin{aligned} \mathcal{I}_{Alice} &= \{\{0\}, \{1,3\}, \{2,4\}, \{5,7\}, \{6,8\}, \{9,11\}, \{10,12\}, \ldots\}, \\ \mathcal{I}_{Bob} &= \{\{0,2\}, \{1\}, \{3,5\}, \{4,6\}, \{7,9\}, \{8,10\}, \{11,13\}, \ldots\}. \end{aligned}$$

Let E be the event $\mathbb{N} \setminus \{0\}$. Then, $K_{Alice}(E) = \mathbb{N} \setminus \{0\}$ and $K_{Bob}(E) = \mathbb{N} \setminus \{0,2\}$, thus $K^1(E) = K(E) = K_{Alice}(E) \cap K_{Bob}(E) = \mathbb{N} \setminus \{0,2\}$. It follows by induction

[3] We recall that the sequence $(K^m(E))_{m>0}$ is *eventually constant* iff there exists $p \geq 0$ such that $K^{p+n}(E) = K^p(E)$, for all $n \geq 0$. It is *strictly shrinking* iff $K^m(E) \subsetneq K^{m+1}(E)$, for all $m \geq 0$.

that $K^m(E) = \mathbb{N} \setminus \{0, 2, \ldots, 2m\}$ for all $m > 0$. Consequently $K^{m+1}(E) \subsetneq K^m(E)$ for all $m > 0$, i.e. the sequence $(K^m(E))_{m>0}$ is strictly shrinking. Moreover, $CK(E) = \bigcap_{m>0} K^m(E) = \bigcap_{m>0} \mathbb{N} \setminus \{0, 2, \ldots, 2m\} = \{2n+1 : n \geq 0\}$. Now, let $L \subseteq \Omega$ be some event different from $CK(E)$, and suppose that the event space $\mathcal{P}(\mathbb{N})$ is equipped with the topology $\mathcal{T} = \{O \subseteq \mathcal{P}(\mathbb{N}) : L \notin O\} \cup \{\mathcal{P}(\mathbb{N})\}$. Then, the only $\mathcal{T}$-open neighbourhood of L is $\mathcal{P}(\mathbb{N})$, and, since all terms of the sequence $(K^m(E))_{m>0}$ are contained in $\mathcal{P}(\mathbb{N})$, it follows that L is a limit point of the sequence $(K^m(E))_{m>0}$. Moreover, L is actually the unique limit point of $(K^m(E))_{m>0}$. Indeed, since $(K^m(E))_{m>0}$ satisfies the strictly shrinking condition, for any event $F \neq L$, the elements of $(K^m(E))_{m>0}$ will never all be contained in the $\mathcal{T}$-open neighbourhood $\{F\}$ of F from some index onwards, showing that F is not a limit point of $(K^m(E))_{m>0}$. Hence, $\lim_{m\to\infty} K^m(E) = L = LK(E)$. Yet since L was precisely chosen to be different from $CK(E)$, it follows that $LK(E) \neq CK(E)$. □

The following example shows that common knowledge may even differ from limit knowledge in the case of so-called well-behaved – i.e. completely metrizable and Hausdorff – topologies.[4]

Example 15. Let $\mathcal{A} = (\mathbb{N}, \mathcal{I}_{Alice}, \mathcal{I}_{Bob})$ be the infinite Aumann structure described in the proof of Proposition 14, and E be the event $\mathbb{N} \setminus \{0\}$. Then, as shown in the proof of Proposition 14, $K^m(E) = \mathbb{N} \setminus \{0, 2, \ldots, 2m\}$ and $CK(E) = \{2n+1 : n \geq 0\}$. Consider farther the Cantor space $\{0,1\}^{\mathbb{N}}$ of functions from $\mathbb{N}$ to $\{0,1\}$ equipped with its usual topology, i.e. the product topology of the discrete topology on $\{0,1\}$. This space is Polish, i.e. Hausdorff and completely metrizable, and the induced metric is defined by $d(f,g) = 2^{-r}$, where $r = \min\{n : f(n) \neq g(n)\}$. Consider finally the sets $F_1 = \{2n : n \geq 0\}$ and $F_2 = \{2n+1 : n \geq 0\}$, and the bijection $f : \mathcal{P}(\mathbb{N}) \longrightarrow \{0,1\}^{\mathbb{N}}$ defined by

$$f(F) = \begin{cases} \chi_F & \text{if } F \neq F_1, F_2 \\ \chi_{F_2} & \text{if } F = F_1 \\ \chi_{F_1} & \text{if } F = F_2\,, \end{cases}$$

where χ_A denotes the characteristic function of A. Now, suppose that the event space $\mathcal{P}(\mathbb{N})$ is equipped with the topology $\mathcal{T}$ defined by letting $O \in \mathcal{T}$ if and only

[4] We recall that a *metric space* (X, d) is a set X together with a metric function $d : X \times X \to \mathbb{R}_+$ satisfying the following properties: for every $x, y, z \in X$,

- $d(x,y) \geq 0$;
- $d(x,y) = 0$ iff $x = y$;
- $d(x,y) = d(y,x)$;
- $d(x,z) \leq d(x,y) + d(y,z)$.

The metric topology on X induced by d is the topology $\mathcal{T}_d$ induced by the open balls of d, i.e., the topology induced by the base

$$\{B(x,r) = \{y \in X : d(x,y) < r\} : \text{for all } x \in X \text{ and } r \in \mathbb{R}_+\}\,.$$

A topological space $(X, \mathcal{T})$ is *metrizable* if there exists a metric $d : X \times X \to \mathbb{R}_+$ whose induced metric topology $\mathcal{T}_d$ coincides with $\mathcal{T}$. In this case, the distance between elements of the space can be explicitly measured via the metric function.
A topological space $(X, \mathcal{T})$ is *completely metrizable* if it is metrizable by some by some metric function d for which every Cauchy sequence is convergent.
A topological space $(X, \mathcal{T})$ is *Hausdorff* (or T_2) iff every distinct points of X have disjoint open neighborhoods.

if $f(O)$ is an open set of the Cantor space. Since f is an homeomorphism from $\mathcal{P}(\mathbb{N})$ to $\{0,1\}^{\mathbb{N}}$, the topological space $(\mathcal{P}(\mathbb{N}), \mathcal{T})$ is also Polish, and hence every sequence converges to at most one limit point. We next prove that the sequence $(\chi_{K^m(E)})_{m>0}$ converges to χ_{F_2} in the Cantor space $\{0,1\}^{\mathbb{N}}$. First of all, the proof of Proposition 14 ensures that $\chi_{K^m(E)} = \chi_{\mathbb{N}\setminus\{0,2,\ldots,2m\}}$ for all $m > 0$. Moreover, observe that $d(\chi_{K^m(E)}, \chi_{F_2}) = 2^{-(m+1)}$. Hence, for any $\varepsilon > 0$, it holds that $d(\chi_{K^m(E)}, \chi_{F_2}) = 2^{-(m+1)} < \varepsilon$ for all $m > \log_2(\frac{1}{\varepsilon}) - 1$. Therefore, $\lim_{m\to\infty} \chi_{K^m(E)} = \chi_{F_2}$. Since f is a homeomorphism, it follows that

$$\begin{aligned} LK(E) = \lim_{m\to\infty} K^m(E) &= \lim_{m\to\infty} f^{-1}(\chi_{K^m(E)}) \\ &= f^{-1}(\lim_{m\to\infty} \chi_{K^m(E)}) = f^{-1}(\chi_{F_2}) = F_1 \neq CK(E), \end{aligned}$$

which yields the desired property. ♣

3.4 Discussion

These considerations show that limit knowledge and common knowledge do differ, and therefore, should not be amalgamated. Even if being both based on the sequence of higher order mutual knowledge, the two operators can be perceived as sharing distinct implicative properties with regards to these claims. Common knowledge bears a standard implicative relation in terms of set inclusion to all iterated mutual knowledge. In contrast, limit knowledge entertains a relation in terms of set proximity with iterated mutual knowledge: the higher the iteration, the closer the respective higher-order mutual knowledge to limit knowledge.

Note that this proximity relation does not comply with any purely logical concept, but relates to a whole variety of possible cognitive perceptions induced by the respective topology. In fact, depending on the considered topology, the induced relation of set proximity between events may reflect physical properties of these events, but also mental representations on the event space. Therefore, as opposed to common knowledge which only captures a single epistemic phenomenon, limit knowledge is able to represent a variety of possible epistemic-topological phenomena as a function on the particular topology on the event space. In this sense, limit knowledge can be viewed as some kind of generalized epistemic-topological concept which, for every possible underlying topology, becomes an operator with precise meaning.

Notably, Proposition 13 ensures that limit knowledge becomes interesting as a possible distinction or refinement of common knowledge precisely in circumstances of the sequence of iterated mutual knowledge being strictly shrinking, i.e., whenever common knowledge actually requires infinitely many higher-order knowledge claims to be computed.

Besides, as opposed to the purely epistemic operator common knowledge, for which factiveness holds, i.e. $CK(E) \subseteq E$, the epistemic-topological operator limit knowledge does in general not bear this property.[5] However, if the sequence of higher-order mutual knowledge of some event E is not strictly shrinking, then limit

[5] For example, in Example 15, an Aumann structure is constructed, the event space is furnished with a topology, and an event E is considered for which $LK(E) \not\subseteq E$.

knowledge is equal to common knowledge, and therefore, limit knowledge is factive, i.e., $LK(E) \subseteq E$. In particular, limit knowledge is factive in all finite Aumann structures.

However, of specific relevance are the situations in which limit knowledge indeed strictly refines common knowledge. In those cases, limit knowledge does imply all iterated mutual knowledge and can be interpreted as some kind of highest iterated mutual knowledge. Example 19 of the next Chapter 4 provides an illustration where limit knowledge is a strict refinement of common knowledge and induces behavioral consequences that cogently differ from the latter.

In general, limit knowledge also differs from approximations of common knowledge such as MONDERER and SAMET (1989)'s common p-belief as well as RUBINSTEIN (1989)'s almost common knowledge. Indeed, common p-belief and almost common knowledge are both implied by common knowledge, whereas limit knowledge is not.

4 LIMIT KNOWLEDGE AND GAMES

4.1 INTRODUCTION

The epistemic operator "common knowledge" has often been argued as being inappropriate for the modelling of a widely shared conception of knowledge. This is mainly due to its standard infinite iterative definition (cf. Definition 8), to its startling implications in AUMANN (1976)'s and MILGROM and STOKEY (1982)'s seminal impossibility agreement and trading results, as well as to a series of so-called common-knowledge paradoxes reviewed by MORRIS (2002). As a consequence, the notion has been challenged and much literature has been focused on the study of game-theoretic consequences of weaker or modified epistemic hypotheses.

For instance, RUBINSTEIN (1989) considered a notion of almost common knowledge and addressed a paradoxical game-theoretic example where the coordination games played under common knowledge and under almost common knowledge substantially differ. LIPMAN (1994) showed that common knowledge of rationality is not, in general, equivalent to the limit of order n mutual knowledge of rationality. Also, MONDERER and SAMET (1989) introduced the notion of common p-belief as a relevant approximation of common knowledge, and BÖRGERS (1994) studied the game theoretic consequences associated with common p-belief. MORRIS (1999) generalized this approach by proposing alternative notions of approximate common knowledge based on the concept of p-belief and studying their consequences in games.

In Chapter 3, we analyzed the relationship between common knowledge and limit knowledge and showed that the two concepts do genuinely differ. Hence, a a natural question concerns the study of the game-theoretic consequences of the epistemic-topological operator limit knowledge. Here, we consider limit knowledge of the specific event rationality and characterize its implications in terms of solution concepts for games. A concrete static infinite game is constructed in which limit knowledge of rationality strictly refines common knowledge of rationality, in terms of solution concepts. In this example, the latter epistemic hypothesis implies iterated strict dominance, while the former entails iterated strict dominance followed by weak dominance. Furthermore, it is generally shown that, for any given game and epistemic model of it satisfying some appropriate epistemic-topological conditions, limit knowledge of rationality is capable of characterizing any possible solution concept. Due to this universal foundational capability, limit knowledge of rationality could thus be used for epistemic-topological characterizations of solution concepts. These considerations argue in favor of a general topological

approach to set-based interactive epistemology.

4.2 Common Knowledge of Rationality

Common knowledge of the particular event that all players are rational has been used in epistemic characterizations of solution concepts in games. A well-known result, e.g., Bernheim (1984), Pearce (1984), Tan and Werlang (1988) as well as Börgers (1993), states that common knowledge of rationality, with the standard notion of rationality as subjective expected utility maximization, provides an epistemic characterization the solution concept "iterated strict dominance". We now give an epistemic foundation of pure strategy iterated strict dominance in terms of common knowledge of some weaker than standard rationality for possibly infinite games. More precisely, we employ a normal form adapted version of Aumann (1995)'s knowledge-based notion of rationality, originally stated for extensive forms with perfect information. As argued by Aumann (1995) and Aumann (1996), this notion is more general and simpler than standard subjective expected utility maximization, since the latter implies the former but the former does not imply the latter, and knowledge-based rationality completely dispenses with probabilities.

Here, the following weaker than standard notion of rationality will be used in the sequel.

Definition 16. Let Γ be a game, $\mathcal{A}^\Gamma$ be an epistemic model of it, and i be some player. The event that i is *rational* is defined as

$$R_i := \bigcap_{s_i \in S_i} (\Omega \setminus K_i(\{\omega \in \Omega : u_i(s_i, \sigma_{-i}(\omega)) > u_i(\sigma(\omega))\})) .$$

Accordingly, a player i is rational – in a weak sense – whenever for any of his strategies $s_i \in S_i$, he does not know that s_i would yield him higher utility than his actual choice. In other words, i is rational at ω if for any of his strategies $s_i \in S_i$ he considers possible a world $\omega' \in \mathcal{I}_i(\omega)$ in which his strategy choice $\sigma_i(\omega')$, being equal to his actual choice $\sigma_i(\omega)$ by measurability, could give him at least as much utility as s_i. The event $R := \bigcap_{i \in I} R_i$ that all players are rational is called *rationality*.

In game theory so-called solution concepts are developed that reduce the strategy profile space. Formally, a solution concept $\mathcal{SC}$ consists of a mapping associating with each game Γ a subset $\mathcal{SC}^\Gamma \subseteq \prod_{i \in I} S_i$ of its strategy profile space. A solution concept thus provides a generic method which does not depend on any particular given game. Intuitively, a solution concept yields the choices a player should make. One of the most established game-theoretic solution concepts for the normal form is iterated strict dominance, which can be defined as follows.

Definition 17. Let $\Gamma = (I, (S_i)_{i \in I}, (u_i)_{i \in I})$ be a game. Moreover, let $S_i^0 = S_i$ for all $i \in I$, and let the sequence $(SD^k)_{k \geq 0}$ of strategy profile sets be inductively given by $SD^0 = \prod_{i \in I} S_i^0$ and $SD^{k+1} = \prod_{i \in I} SD_i^{k+1}$, where $SD_i^{k+1} = SD_i^k \setminus \{s_i \in SD_i^k :$ there exists $s_i' \in SD_i^k$ such that $u_i(s_i, s_{-i}) < u_i(s_i', s_{-i})$, for all $s_{-i} \in SD_{-i}^k\}$, for all $i \in I$. The solution concept *iterated strict dominance* is defined as $ISD^\Gamma := \bigcap_{k \geq 0} SD^k$.

Note that Dufwenberg and Stegeman (2002) study iterated strict dominance for arbitrary static games in a non-epistemic context, unveiling potential ill-bevaiour.

It is shown that iterated strict dominance can be order-dependent, return an empty set of strategy profiles, or fail to yield a maximal reduction after countably many steps. Moreover, they prove the existence and uniqueness of a non-empty maximal reduction by requiring compactness of the players' strategy spaces and continuity of the utility functions. However, according to Definition 17, order dependence is no longer a possible problem, since at each round, all the remaining strictly dominated strategies are eliminated.

We now give an epistemic foundation of pure strategy iterated strict dominance in terms of common knowledge of rationality for possibly infinite games with the weaker than standard concept of knowledge-based rationality. Note that in Proposition 18 below, as well as in all results of Section 4.3, common knowledge of the structure of the game is endorsed as an implicit background assumption.

Proposition 18. *Let Γ be a game and $\mathcal{A}^\Gamma$ be an epistemic model of it. Then, $\sigma(CK(R)) \subseteq ISD^\Gamma$.*

Proof. By induction, we prove that $\sigma(K^m(R)) \subseteq SD^{m+1}$, for all $m \geq 0$. It then follows that $\sigma(CK(R)) = \sigma(\bigcap_{m>0} K^m(R)) = \sigma(\bigcap_{m\geq 0} K^m(R)) \subseteq \bigcap_{m\geq 0} \sigma(K^m(R)) \subseteq \bigcap_{m\geq 0} SD^{m+1} = \bigcap_{m>0} SD^m = \bigcap_{m\geq 0} SD^m = ISD^\Gamma$. First of all, consider $(s_i)_{i\in I} \in \sigma(K^0(R)) = \sigma(R)$. Then, there exists $\omega \in R = \bigcap_{i\in I} R_i$ such that $\sigma(\omega) = (s_i)_{i\in I}$. Hence, by definition of R_i and measurability of σ_i, for all $s_i \in S_i$, there exists $\omega' \in \mathcal{I}_i(\omega)$ such that $u_i(s_i, \sigma_{-i}(\omega')) \leq u_i(\sigma(\omega')) = u_i(\sigma_i(\omega), \sigma_{-i}(\omega'))$. It follows that $\sigma_i(\omega) \in SD_i^1$ for all $i \in I$, and thus $\sigma(\omega) = (s_i)_{i\in I} \in \prod_{i\in I} SD_i^1 = SD^1$. Therefore, $\sigma(K^0(R)) \subseteq SD^1$ obtains. Now, assume $\sigma(K^m(R)) \subseteq SD^{m+1}$ for some $m > 0$, and let $(s_i)_{i\in I} \in \sigma(K^{m+1}(R))$. Then, there exists $\omega \in K^{m+1}(R)$ such that $\sigma(\omega) = (s_i)_{i\in I}$. Hence, $\mathcal{I}_i(\omega) \subseteq K^m(R)$, and thus, by the induction hypothesis, $\sigma(\mathcal{I}_i(\omega)) \subseteq SD^{m+1}$ obtains. Besides, since $\omega \in R_i$, for all $s_i \in SD_i^{m+1}$ there exists $\omega' \in \mathcal{I}_i(\omega)$ such that $u_i(s_i, \sigma_{-i}(\omega')) \leq u_i(\sigma(\omega')) = u_i(\sigma_i(\omega), \sigma_{-i}(\omega'))$. Yet since $\sigma(\mathcal{I}_i(\omega)) \subseteq SD^{m+1}$, each $\omega' \in \mathcal{I}_i(\omega)$ induces $\sigma_{-i}(\omega') \in SD_{-i}^{m+1}$, which in turn implies that $\sigma_i(\omega) \in SD_i^{m+2}$ for all $i \in I$, and consequently $(s_i)_{i\in I} = \sigma(\omega) \in \prod_{i\in I} SD_i^{m+2} = SD^{m+2}$. Therefore, $\sigma(K^{m+1}(R)) \subseteq SD^{m+2}$ holds, which concludes the proof. □

4.3 Limit Knowledge of Rationality

The new epistemic operator limit knowledge can be used in the context of games. Indeed, we now illustrate that limit knowledge is capable of refining common knowledge in terms of solution concepts. More precisely, a Cournot-type game is constructed where the application of iterated strict dominance followed by weak dominance, denoted by $(ISD + WD)^\Gamma$ for a given game Γ, is a strict refinement of iterated strict dominance.[1] Then, an epistemic Aumann model of this game is given such that the event common knowledge of rationality precisely reveals all the possible strategy profiles that survive iterated strict dominance, while limit knowledge of rationality conveys the unique strategy profile in accordance with iterated strict dominance followed by weak dominance. Moreover, in this case, limit knowledge

[1] Formally, given a game Γ, iterated strict dominance followed by weak dominance is defined as $(ISD + WD)^\Gamma := \prod_{i\in I}(ISD_i^\Gamma \setminus \{s_i \in ISD_i^\Gamma : \text{there exists } s_i' \in ISD_i^\Gamma \text{ such that } u_i(s_i, s_{-i}) \leq u_i(s_i', s_{-i}) \text{ for all } s_{-i} \in ISD_{-i}^\Gamma \text{ and } u_i(s_i, s_{-i}') < u_i(s_i', s_{-i}') \text{ for some } s_{-i}' \in ISD_{-i}^\Gamma\})$.

of rationality being strictly included in common knowledge of rationality is thus being interpretable as some kind of highest iterated mutual knowledge.

Example 19. Consider the Cournot-type game $\Gamma = (I, (S_i)_{i\in I}, (u_i)_{i\in I})$ with player set $I = \{Alice, Bob, Claire, Donald\}$, strategy sets $S_{Alice} = S_{Bob} = [0,1]$, $S_{Claire} = \{U, D\}$, $S_{Donald} = \{L, R\}$, and utility functions $u_i : S_{Alice} \times S_{Bob} \times S_{Claire} \times S_{Donald} \to \mathbb{R}$, for all $i \in I$, defined by $u_{Alice}(x, y, v, w) = x(1 - x - y)$, $u_{Bob}(x, y, v, w) = y(1 - x - y)$, and $u_{Claire}(x, y, v, w)$ and $u_{Donald}(x, y, v, w)$ as given in Figure 4.

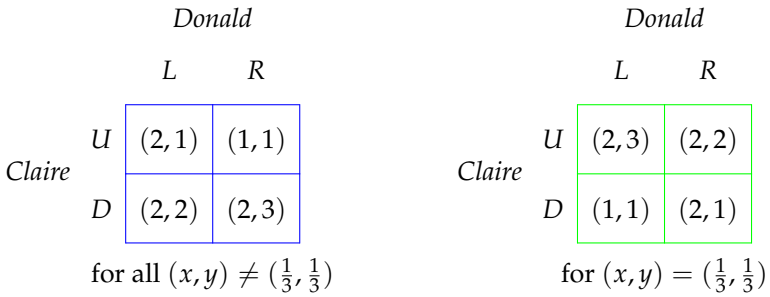


Figure 4 – The four player game of Example 19.

Solving the game by iterated strict dominance – requiring infinitely many elimination rounds – yields the sequence of successively surviving strategy profile sets $\left([a_n, b_n]^2 \times \{U, D\} \times \{L, R\}\right)_{n\geq 0}$, where $[a_0, b_0] = [0, 1]$, $[a_{n+1}, b_{n+1}] = [\frac{a_n+b_n}{2}, b_n]$ if n is odd, and $[a_{n+1}, b_{n+1}] = [a_n, \frac{a_n+b_n}{2}]$ if n is even. The non-unique solution of the game is thus given by the remaining four strategy profiles

$$ISD^\Gamma = \bigcap_{n\geq 0} \left([a_n, b_n]^2 \times \{U, D\} \times \{L, R\}\right) = \{\frac{1}{3}\} \times \{\frac{1}{3}\} \times \{U, D\} \times \{L, R\}.$$

However, it is possible to further restrict the remaining strategy sets of *Claire* and *Donald* by a weak dominance argument – a potential refinement that only emerges *after* applying iterated strict dominance. Indeed, in the set ISD^Γ the strategies D and R are weakly dominated by U and L for *Claire* and *Donald*, respectively. Therefore, iterated strict dominance followed by weak dominance yields the singleton set $(ISD + WD)^\Gamma = \{(\frac{1}{3}, \frac{1}{3}, U, L)\}$ as a possible strictly refined solution of the game.

Before turning towards the epistemic Aumann model of this game, some preliminary observations are needed. Note that, by definition of the utility functions, the best response strategy of *Alice* to an opponents' strategy combination only depends on *Bob*'s choice, and vice versa. More precisely, *Alice*'s and *Bob*'s best response functions $b_{Alice} : [0,1] \times \{U, D\} \times \{L, R\} \to [0,1]$ and $b_{Bob} : [0,1] \times \{U, D\} \times \{L, R\} \to [0,1]$ are given by $b_{Alice}(y, v, w) = \frac{1-y}{2}$ and $b_{Bob}(x, v, w) = \frac{1-x}{2}$, respectively. Hence, the set of all strategy profiles in which *Alice* and *Bob* simultaneously play best responses is given by $\{\frac{1}{3}\} \times \{\frac{1}{3}\} \times \{U, D\} \times \{L, R\}$. On the basis of these two functions, we now describe an infinite sequence $(s^n_{Alice}, s^n_{Bob})_{n\geq 0}$ of strategy combinations for *Alice* and *Bob* which will be central to the construction of our

epistemic Aumann model. This sequence is defined for all $n \geq 0$ by induction as follows.

$$\begin{aligned}
\left(s^0_{Alice}, s^0_{Bob}\right) &= (0,1) \\
\left(s^1_{Alice}, s^1_{Bob}\right) &= \left(0, \frac{1}{2}\right) \\
\left(s^{2n+2}_{Alice}, s^{2n+2}_{Bob}\right) &= \left(\frac{1 - s^{2n+1}_{Bob}}{2}, s^{2n+1}_{Bob}\right) \\
\left(s^{2n+3}_{Alice}, s^{2n+3}_{Bob}\right) &= \left(s^{2n+2}_{Alice}, \frac{1 - s^{2n+2}_{Alice}}{2}\right)
\end{aligned}$$

This infinite sequence $(s^n_{Alice}, s^n_{Bob})_{n\geq 0}$ of strategy combinations for *Alice* and *Bob* is illustrated in Figure 5. The depicted points indicate its first few elements. Note that the terms $\left(s^{2n+2}_{Alice}, s^{2n+2}_{Bob}\right)$ and $\left(s^{2n+3}_{Alice}, s^{2n+3}_{Bob}\right)$ are given by the projections of their predecessors on *Alice*'s and *Bob*'s best response curves, respectively. Farther observe that the sequence converges to $(\frac{1}{3}, \frac{1}{3})$.

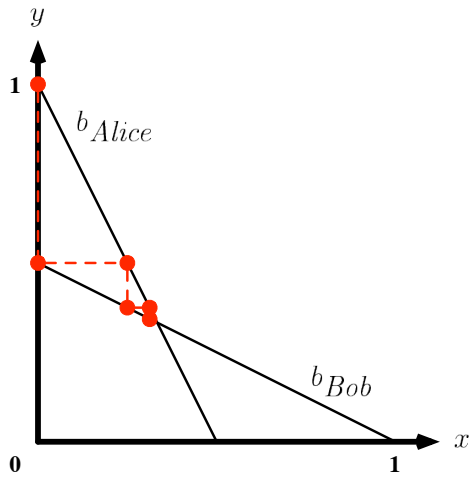


Figure 5 – Illustration of the infinite sequence $(s^n_{Alice}, s^n_{Bob})_{n\geq 0}$ of strategy combinations for *Alice* and *Bob*.

Next, consider the epistemic Aumann model $\mathcal{A}^\Gamma = (\Omega, (\mathcal{I}_i)_{i\in I}, (\sigma_i)_{i\in I})$ of Γ, where the countable set of all possible worlds is given by

$$\Omega = \{\alpha, \beta, \gamma, \delta, \alpha_0, \beta_0, \gamma_0, \delta_0, \alpha_1, \beta_1, \gamma_1, \delta_1, \alpha_2, \beta_2, \gamma_2, \delta_2, \ldots\},$$

the players' information partitions are specified by

$$\mathcal{I}_{Alice} = \{\{\alpha, \beta, \gamma, \delta\}\} \cup \{\{\alpha_{2n}, \beta_{2n}, \gamma_{2n}, \delta_{2n}, \alpha_{2n+1}, \beta_{2n+1}, \gamma_{2n+1}, \delta_{2n+1}\} : n \geq 0\},$$

$$\mathcal{I}_{Bob} = \{\{\alpha, \beta, \gamma, \delta\}, \{\alpha_0, \beta_0, \gamma_0, \delta_0\}\} \cup \{\{\alpha_{2n-1}, \beta_{2n-1}, \gamma_{2n-1}, \delta_{2n-1}, \alpha_{2n}, \beta_{2n}, \gamma_{2n}, \delta_{2n}\} : n > 0\},$$

$$\mathcal{I}_{Claire} = \{\{\alpha, \beta\}, \{\gamma, \delta\}\} \cup \{\{\alpha_n, \beta_n\} : n \geq 0\} \cup \{\{\gamma_n, \delta_n\} : n \geq 0\},$$

$$\mathcal{I}_{Donald} = \{\{\alpha, \gamma\}, \{\beta, \delta\}\} \cup \{\{\alpha_n, \gamma_n\} : n \geq 0\} \cup \{\{\beta_n, \delta_n\} : n \geq 0\},$$

and the choice function $\sigma = (\sigma_{Alice}, \sigma_{Bob}, \sigma_{Claire}, \sigma_{Donald}) : \Omega \to \prod_{i \in I} S_i$ assembling all the players' choice functions is defined for all $n \geq 0$ by:

$$\begin{aligned} \sigma(\alpha) &= (1/3, 1/3, U, L) & \sigma(\alpha_n) &= (s^n_{Alice}, s^n_{Bob}, U, L) \\ \sigma(\beta) &= (1/3, 1/3, U, R) & \sigma(\beta_n) &= (s^n_{Alice}, s^n_{Bob}, U, R) \\ \sigma(\gamma) &= (1/3, 1/3, D, L) & \sigma(\gamma_n) &= (s^n_{Alice}, s^n_{Bob}, D, L) \\ \sigma(\delta) &= (1/3, 1/3, D, R) & \sigma(\delta_n) &= (s^n_{Alice}, s^n_{Bob}, D, R). \end{aligned}$$

By definition of the sequence $(s^n_{Alice}, s^n_{Bob})_{n \geq 0}$, the two equalities $s^{2n}_{Alice} = s^{2n+1}_{Alice}$ and $s^{2n+1}_{Bob} = s^{2n+2}_{Bob}$ hold for all $n \geq 0$, and therefore our epistemic Aumann model satisfies the standard measurability requirement for the players' choice functions.

We now analyze the players' rationality. First of all, consider *Alice* and note that she is rational at worlds α, β, γ and δ. By construction of the sequence $(s^n_{Alice}, s^n_{Bob})_{n \geq 0}$, if ω is a world such that $(\sigma_{Alice}(\omega), \sigma_{Bob}(\omega)) = (s^{2n}_{Alice}, s^{2n}_{Bob})$ for some $n \geq 0$, then $u_{Alice}(\sigma(\omega)) = u_{Alice}(b_{Alice}(\sigma_{-Alice}(\omega)), \sigma_{-Alice}(\omega)) \geq u_{Alice}(x, \sigma_{-Alice}(\omega))$, for all $x \in S_{Alice}$. It follows that *Alice* is rational at every world $\omega' \in \mathcal{I}_{Alice}(\omega)$. By definition of $\mathcal{I}_{Alice}$, since each cell contains a world ω such that $(\sigma_{Alice}(\omega), \sigma_{Bob}(\omega)) = (s^{2n}_{Alice}, s^{2n}_{Bob})$ for some $n \geq 0$, it follows that $R_{Alice} = \Omega$. Secondly, *Bob* is shown not to be rational at every possible world. In fact, his strategies $\sigma_{Bob}(\alpha_0), \sigma_{Bob}(\beta_0), \sigma_{Bob}(\gamma_0)$ and $\sigma_{Bob}(\delta_0)$ all equal 1, which in turn is strictly dominated by any $y \in (0, 1)$, thus $\alpha_0, \beta_0\gamma_0, \delta_0 \notin R_{Bob}$. Analogous reasoning as for *Alice* allows to conclude that *Bob* is rational at all remaining worlds. Therefore, $R_{Bob} = \Omega \setminus \{\alpha_0, \beta_0, \gamma_0, \delta_0\}$. Finally, *Claire* and *Donald* are rational at every possible world. Indeed, observe that *Claire* is rational at α, since $\alpha \in \mathcal{I}_{Claire}(\alpha)$ and $u_{Claire}(\sigma(\alpha)) \geq u_{Claire}(D, \sigma_{-Claire}(\alpha))$, where D is her only alternative strategy. As $\beta \in \mathcal{I}_{Claire}(\alpha)$, it follows that *Claire* is also rational at β. Similar arguments hold for *Claire*'s rationality at worlds γ and δ. Analogously, *Claire* is rational at all other possible worlds α_n, β_n, γ_n and δ_n, for all $n \geq 0$. *Donald*'s rationality at each world is obtained in the same manner. Therefore, $R_{Claire} = R_{Donald} = \Omega$ and the event of all players being rational is given by $R = \bigcap_{i \in I} R_i = \Omega \setminus \{\alpha_0, \beta_0, \gamma_0, \delta_0\}$.

It follows that $K(R) = \bigcap_{i \in I} K_i(R) = \Omega \setminus \{\alpha_0, \beta_0, \gamma_0, \delta_0, \alpha_1, \beta_1, \gamma_1, \delta_1\}$, and by induction $K^m(R) = \Omega \setminus \{\alpha_0, \beta_0, \gamma_0, \delta_0, \alpha_1, \beta_1, \gamma_1, \delta_1, \ldots, \alpha_m, \beta_m, \gamma_m, \delta_m\}$ for all $m > 0$. Therefore, $CK(R) = \bigcap_{m > 0} K^m(R) = \{\alpha, \beta, \gamma, \delta\}$. Besides, suppose the event space $\mathcal{P}(\Omega)$ to be equipped with the topology $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : \{\alpha\} \notin O\} \cup \{\mathcal{P}(\Omega)\}$.

Then, the only $\mathcal{T}$-open neighbourhood of the event $\{\alpha\}$ is $\mathcal{P}(\Omega)$, and all terms of the sequence $(K^m(R))_{m>0}$ are contained in $\mathcal{P}(\Omega)$. Thus $(K^m(R))_{m>0}$ converges to $\{\alpha\}$. Moreover, any singleton $\{F\} \neq \{\{\alpha\}\}$ is $\mathcal{T}$-open, and, since $K^{m+1}(R) \subsetneq K^m(R)$ for all $m > 0$, the strictly shrinking sequence $(K^m(R))_{m>0}$ will never remain in the open neighbourhood $\{F\}$ of F from some index onwards. Hence, $(K^m(R))_{m>0}$ does not converge to any such event F. Therefore the limit $(K^m(R))_{m>0}$ is unique, and $LK(R) = \lim_{m\to\infty}(K^m(R))_{m>0} = \{\alpha\}$.

Farther, $\sigma(CK(R)) = \{\sigma(\alpha), \sigma(\beta), \sigma(\gamma), \sigma(\delta)\} = \{\frac{1}{3}\} \times \{\frac{1}{3}\} \times \{U, D\} \times \{L, R\} = ISD^\Gamma$, while $\sigma(LK(R)) = \{\sigma(\alpha)\} = \{(\frac{1}{3}, \frac{1}{3}, U, L)\} = (ISD + WD)^\Gamma$. Hence, the solution in accordance with $LK(R)$ is a strict refinement of the solution induced by $CK(R)$. ♣

The preceding example describes a particular epistemic-topological epistemic model of a given game such that limit knowledge of rationality is a strict refinement of common knowledge of rationality in terms of solution concepts.

More generally, we now show that for any game and epistemic Aumann model of it such that the sequence of iterated mutual knowledge of rationality is strictly shrinking, every solution concept can be epistemic-topologically characterized by limit knowledge of rationality.

Theorem 20. *Let Γ be a game, $\mathcal{A}^\Gamma$ be an epistemic Aumann model of it such that the sequence $(K^m(R))_{m>0}$ is strictly shrinking, and $\mathcal{SC}$ be some solution concept. Then, there exists a topology on $\mathcal{P}(\Omega)$ such that $\sigma(LK(R)) \subseteq \mathcal{SC}^\Gamma$.*

Proof. Consider the event $F = \sigma^{-1}(\mathcal{SC}^\Gamma) = \{\omega \in \Omega : \sigma(\omega) \in \mathcal{SC}^\Gamma\}$. Then $\sigma(F) \subseteq \mathcal{SC}^\Gamma$. Now, suppose the event space $\mathcal{P}(\Omega)$ to be equipped with the topology $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : F \notin O\} \cup \{\mathcal{P}(\Omega)\}$. By definition of $\mathcal{T}$, the only $\mathcal{T}$-open neighbourhood of F is $\mathcal{P}(\Omega)$, and thus the sequence $(K^m(R))_{m>0}$ converges to F. Besides, for every event $E \neq F$, the singleton $\{E\}$ is open, and, since satisfying the strictly shrinking condition, $(K^m(R))_{m>0}$ will never remain in the $\mathcal{T}$-open neighbourhood $\{E\}$ of E from some index onwards. Consequently, $(K^m(R))_{m>0}$ does not converge to E. It follows that the limit of $(K^m(R))_{m>0}$ is unique, and $LK(R) = \lim_{m\to\infty}(K^m(R))_{m>0} = F$. Therefore, $\sigma(LK(R)) = \sigma(F) \subseteq \mathcal{SC}^\Gamma$. □

4.4 Discussion

We studied some game-theoretic consequences of limit knowledge, and showed that this epistemic-topological operator is capable of relevant characterizations of solution concepts in games. In fact, we provided a concrete static infinite game in which limit knowledge of rationality strictly refines common knowledge of rationality in terms of solution concepts (Example 19). Furthermore, we showed that limit knowledge of rationality can provide an epistemic-topological foundation for any possible game-theoretic solution concept (Theorem 20).

Note that this universal characterization property is enabled via the choice of a specific topology on the event space constructed for the purpose to be achieved.[2]

[2] In fact, in the proof of Theorem 20, we used the so-called "excluded point topology" to make the sequence of higher-order mutual knowledge claims converge to the specific desired event.

Hence, a natural approach to be followed concerns the epistemic-topological foundations of solution concepts obtained on the basis topologies on the event space that are rather "epistemically-plausible": for instance, those revealing some kind of underlying agent perceptions or reasoning patterns, as well as natural extensions of implicit topologies on the state space.

As an example, a plausible epistemic-topological foundation for the solution concept k-times strict dominance SD^k is given now. Suppose a game in normal form Γ and some epistemic model $\mathcal{A}^\Gamma$ of it such that the sequence $(K^m(R))_{m>0}$ is strictly shrinking. Consider the topology $\mathcal{T}$ on $\mathcal{P}(\Omega)$ induced by the subbase

$$\begin{aligned}&\{\{K^m(R) : m > 0\}, \mathcal{P}(\Omega) \setminus \{K^m(R) : m > 0\}\}\\ &\cup \{\{K^m(R)\} : m < k-1\}\\ &\cup \left\{\{K^k(R), K^{k+1}(R), \ldots, K^n(R)\} : n > k-1\right\}.\end{aligned}$$

This topology can be argued to be epistemically plausible in the sense that it satisfies the following four properties.

1. If E is a term of the sequence $(K^m(R))_{m>0}$ and F is not (or vice versa), then E and F are T_2-separable.[3]
2. If E and F are two distinct terms of $(K^m(R))_{m>0}$ of index strictly smaller than $k-1$, then E and F are T_2-separable.
3. If E and F are two distinct terms of $(K^m(R))_{m>0}$ of index strictly larger than $k-1$, then E and F are T_0-separable.[4]
4. If $E = K^{k-1}(R)$ and F is another term of $(K^m(R))_{m>0}$ (or vice versa), then E and F are T_0-separable.

These properties reflect a particular perception of the event space, where the agents' topological distinction between the first $(k-2)$-order knowledge of rationality is stronger than between the remaining higher-order mutual knowledge. By definition of $\mathcal{T}$, it follows that $LK(R) = K^{k-1}(R)$ and therefore the proof of Proposition 18 ensures that $\sigma(LK(R)) \subseteq SD^k$. In this sense, $\mathcal{T}$ provides a plausible epistemic-topological foundation for the solution concept SD^k.

These considerations show that the specific hypothesis "limit knowledge of rationality" is indeed capable of modelling agents' reasoning patterns involving some notion of closeness between events, and in turn, of characterizing relevant solution concepts in games. More generally, analogous to the epistemic program in game theory that attempts to provide epistemic foundations for solution concepts, the proposed epistemic-topological approach to game theory intends to unveil relevant epistemic-topological foundations of existing and novel solution concepts.

[3]Given a topological space $(X, \mathcal{T})$ two points $a, b \in X$ are called T_2-separable if there exist two disjoint $\mathcal{T}$-neighbourhoods N_a and N_b of a and b, respectively.

[4]Given a topological space $(X, \mathcal{T})$ two points in X are called T_0-separable if there exists an $\mathcal{T}$-open set containing precisely one of the two points.

5 Limit Knowledge and Aumann's Agreement Theorem

5.1 Introduction

Aumann (1976)'s so-called agreement theorem establish the impossibility for two agents to agree to disagree on their posterior beliefs. More precisely, if two Bayesian agents are equipped with a common prior belief, receive private information, and have common knowledge of their posterior beliefs, then these posteriors must be equal. In other words, among Bayesian agents with a common prior, distinct posteriors cannot be common knowledge. In this sense, agents cannot agree to disagree.

Along these lines, Milgrom and Stokey (1982) deduced an impossibility theorem of speculative trade. Intuitively, their result states that if two traders agree on a prior efficient allocation of goods, then upon receiving private information, it cannot be common knowledge that both traders have an incentive to trade.

From an empirical or quasi-empirical point of view, these impossibility results are rather startling since real world agents do frequently disagree on a large variety of issues. Consequently, the notion of common knowledge has been challenged and much literature has been focused on reconsidering Aumann (1976)'s agreement theorem in light of weakened or modified epistemic assumptions.

In this spirit, Geanakoplos and Polemarchakis (1982) showed that without assuming common knowledge of the posteriors, agents following a specific communication procedure can nevertheless not agree to disagree. A different generalization was provided by Bacharach (1985)'s non-probabilistic agreement theorem. The result states that if two agents follow a common decision procedure in line with the sure thing principle (i.e., for every event and every partition of it, whenever each cell of the partition induces a same decision, the event itself generates precisely this decision) and if their particular decisions are common knowledge, then these decisions must coincide. Besides, Samet (1990) adopted a bounded rationality approach of the problem. He dropped the implicit negative introspection assumption (which states that for every proposition that an agent does not know, the agent actually knows that he does not know it) and established that Aumann's agreement theorem remains valid with agents ignorant of their own ignorance.

In addition, Monderer and Samet (1989) introduced the notion of common p-belief as a relevant weakening of common knowledge, and proved that Aumann's result on the impossibility of agreeing to disagree can be approximated in this con-

text. They showed that if Bayesian agents equipped with a common prior have common p-belief of their posteriors for a sufficiently large p, then these posteriors cannot differ significantly. This result was further generalized by NEEMAN (1996) and KAJII and MORRIS (1997). Moreover, SONSINO (1995) and NEEMAN (1996) independently proved that the impossibility of speculative trade does actually not hold anymore under the epistemic hypotheses of almost common knowledge and common p-belief, if sufficiently large noise is considered. MORRIS (1999) generalized this approach by proposing alternative notions of approximate common knowledge based on the concept of p-belief and studying their consequences in the contexts agreeing to disagree and no trade results.

More recently, BACH and PEREA (2013) showed that Aumann's impossibility result is not robust with respect to the common prior assumption in the sense that two Bayesian agents with arbitrarily close prior beliefs can have common knowledge of completely opposed posteriors. On that basis, they established a lexicographic agreement theorem. Besides, the agreement theorem has also been analyzed from the perspective of dynamic epistemic logic. DÉGREMONT and ROY (2012) obtained a non-probabilistic impossibility result with common belief – instead of common knowledge – of posteriors within the framework of epistemic plausibility models, when the common priors satisfy a specific well-foundedness assumption. Moreover, several probabilistic agreement theorems are established by DEMEY (2014) using enriched probabilistic Kripke models.

For deeper considerations about Aumann's agreement theorem, see the comprehensive surveys by BONANNO and NEHRING (1997) as well as by MÉNAGER (2012).

Here, we analyze agreeing to disagree in a topologically extended epistemic framework. The epistemic operator common knowledge is replaced by the epistemic-topological operator limit knowledge. We show that Bayesian agent entertaining common priors as well as limit knowledge of their posteriors can have these posteriors being distinct. Since limit knowledge is defined as the topological limit of higher-order mutual knowledge, our result can be interpreted as establishing – in contrast to Aumann's impossibility theorem – that agents can agree to disagree, or more precisely, can limit-agree to disagree. These results show that Aumann's agreement theorem is not robust when considered from a more general epistemic-topological perspective, with common knowledge being replaced by limit knowledge. We further show that the notion of limit knowledge is able to capture relevant reasoning patterns for which agents do agree to disagree on their posterior beliefs. In particular, an example is constructed in which limit knowledge is identical with RUBINSTEIN (1989)'s notion of almost common knowledge, i.e., with m iterations of mutual knowledge for some finite number m. Thereby, we give an epistemic-topological foundation for RUBINSTEIN (1989)'s notion of almost common knowledge. In this specific case also, agents can limit-agree to disagree. Once again, these consideration argue in favor of a general topological framework for interactive epistemology.

5.2 AUMANN'S AGREEING TO DISAGREE

Aumann's agreement theorem states that if two agents have a common prior and their posterior beliefs in some event are common knowledge, then these posterior beliefs must coincide. In other words, if two agents with common prior beliefs hold distinct posterior beliefs, then these posterior beliefs cannot be common knowledge among them. Intuitively, it is impossible for agents to consent to distinct beliefs. Thus, agents cannot agree to disagree.

Aumann's result is now formalized in a slightly modified way. More precisely, the introduction of arbitrary values for the agents' posterior beliefs as in Aumann's original statement is dispensed with. Instead, these arbitrary values are substituted by conceivable posterior beliefs, namely updated prior beliefs induced by some auxiliary world. The values of the posteriors are thus made endogenous, as nothing external to the formal structure is needed for their determination. In this sense, the following statement of the agreement theorem is inherent to the formal structure it is embedded in.

Aumann's Agreement Theorem (Version 1)**.** *Let $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i=1}^n, p)$ be an Aumann structure, $E \subseteq \Omega$ be an event, and $\hat{\omega}$ be a world. If $CK(\bigcap_{i=1}^n \{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}) \neq \emptyset$, then $p(E \mid \mathcal{I}_1(\hat{\omega})) = p(E \mid \mathcal{I}_2(\hat{\omega})) = \cdots = p(E \mid \mathcal{I}_n(\hat{\omega}))$.*

Proof. Let $\hat{\omega} \in \Omega$ be such that $CK(\bigcap_{i=1}^n \{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}) \neq \emptyset$, and for sake of notational convenience let the event $\bigcap_{i=1}^n \{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}$ be denoted by E'. Consider a world $\omega \in CK(E')$ and some agent $i^* \in \{1, 2, \ldots, n\}$. As the meet $\bigwedge_{i=1}^n \mathcal{I}_i$ is coarser than agent's i^* possibility partition, the cell $\bigwedge_{i=1}^n \mathcal{I}_i(\omega)$ can be written as a disjoint union of information cells of i^*, namely there exists a set $A_{i^*} \subseteq \Omega$ such that $\bigwedge_{i=1}^n \mathcal{I}_i(\omega) = \bigcup_{\omega' \in A_{i^*}} \mathcal{I}_{i^*}(\omega')$, and for all $\omega_1, \omega_2 \in A_{i^*}$, if $\omega_1 \neq \omega_2$, then $\mathcal{I}_{i^*}(\omega_1) \neq \mathcal{I}_{i^*}(\omega_2)$. Since $\omega \in CK(E')$, the meet definition of common knowledge ensures that $\bigwedge_{i=1}^n \mathcal{I}_i(\omega) \subseteq E'$. By the definition of the event E' it then follows that $p(E \mid \mathcal{I}_{i^*}(\hat{\omega})) = p(E \mid \mathcal{I}_{i^*}(\omega'))$, for all $\omega' \in A_{i^*}$. Thus, $p(E \cap \mathcal{I}_{i^*}(\omega')) = p(E \mid \mathcal{I}_{i^*}(\hat{\omega})) \cdot p(\mathcal{I}_{i^*}(\omega'))$, for all $\omega' \in A_{i^*}$. Summing over the worlds in A_{i^*} yields the following equation of sums $\sum_{\omega' \in A_{i^*}} p(E \cap \mathcal{I}_{i^*}(\omega')) = p(E \mid \mathcal{I}_{i^*}(\hat{\omega})) \cdot \sum_{\omega' \in A_{i^*}} p(\mathcal{I}_{i^*}(\omega'))$. By countable additivity of the probability measure p, pairwise disjointness of the events $E \cap \mathcal{I}_{i^*}(\omega')$ for all $\omega' \in A_{i^*}$, and distributivity of intersection, it follows that $\sum_{\omega' \in A_{i^*}} p(E \cap \mathcal{I}_{i^*}(\omega')) = p(\bigcup_{\omega' \in A_{i^*}} (E \cap \mathcal{I}_{i^*}(\omega'))) = p(E \cap \bigcup_{\omega' \in A_{i^*}} \mathcal{I}_{i^*}(\omega')) = p(E \cap \bigwedge_{i=1}^n \mathcal{I}_i(\omega))$ and $\sum_{\omega' \in A_{i^*}} p(\mathcal{I}_{i^*}(\omega')) = p(\bigcup_{\omega' \in A_{i^*}} \mathcal{I}_{i^*}(\omega')) = p(\bigwedge_{i=1}^n \mathcal{I}_i(\omega))$. The equation of sums can then be written as $p(E \cap \bigwedge_{i=1}^n \mathcal{I}_i(\omega)) = p(E \mid \mathcal{I}_{i^*}(\hat{\omega})) \cdot p(\bigwedge_{i=1}^n \mathcal{I}_i(\omega))$, thence $p(E \mid \mathcal{I}_{i^*}(\hat{\omega})) = p(E \cap \bigwedge_{i=1}^n \mathcal{I}_i(\omega)) / p(\bigwedge_{i=1}^n \mathcal{I}_i(\omega))$. Since i^* has been arbitrarily chosen, the latter equality holds for every agent $i \in \{1, 2, \ldots, n\}$. Therefore, $p(E \mid \mathcal{I}_1(\hat{\omega})) = p(E \mid \mathcal{I}_2(\hat{\omega})) = \cdots = p(E \mid \mathcal{I}_n(\hat{\omega})) = p(E \cap \bigwedge_{i=1}^n \mathcal{I}_i(\omega)) / p(\bigwedge_{i=1}^n \mathcal{I}_i(\omega))$, which concludes the proof. □

The previous theorem states that, if common knowledge of the agents' posterior beliefs being equal to the values induced by some auxiliary world is non-empty, then the agents' posterior beliefs at this given world coincide. Yet as a consequence, the agents' posterior beliefs do not only coincide at the auxiliary world, but also at every possible world inducing the same values as the auxiliary world. In particular,

every world contained in common knowledge of the agents' posterior beliefs satisfies equality of posterior beliefs. Thus, a second version of Aumann's agreement theorem ensues as follows.

Aumann's Agreement Theorem (Version 2). *Let $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i=1}^n, p)$ be an Aumann structure, $E \subseteq \Omega$ be an event, and $\hat{\omega}, \omega \in \Omega$ be worlds such that $CK(\bigcap_{i=1}^n \{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}) \neq \emptyset$ and $\omega \in CK(\bigcap_{i=1}^n \{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\})$. Then, $p(E \mid \mathcal{I}_1(\omega)) = p(E \mid \mathcal{I}_2(\omega)) = \cdots = p(E \mid \mathcal{I}_n(\omega))$.*

Proof. Again, for sake of notational convenience, let the event $\bigcap_{i=1}^n \{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}$ be denoted by E'. Since $\omega \in CK(E')$, it holds that $\omega \in E'$. Observe that the definition of the event E' ensures that $p(E \mid \mathcal{I}_i(\omega)) = p(E \mid \mathcal{I}_i(\hat{\omega}))$ for all agents $i \in \{1, 2, \ldots, n\}$. Also, Theorem 5.2 implies that $p(E \mid \mathcal{I}_1(\hat{\omega})) = p(E \mid \mathcal{I}_2(\hat{\omega})) = \cdots = p(E \mid \mathcal{I}_n(\hat{\omega}))$. Therefore, $p(E \mid \mathcal{I}_1(\omega)) = p(E \mid \mathcal{I}_2(\omega)) = \cdots = p(E \mid \mathcal{I}_n(\omega))$. □

The two preceding versions of Aumann's Agreement Theorem provide slightly different formulations of the impossibility for agents to agree to disagree on their posterior beliefs. In the first version, emphasis is put on the posterior beliefs induced by some auxiliary world, whereas the second version focuses on the posterior beliefs that the agents actually hold. Basically, the first version intuitively states that, if the agents' posterior beliefs are common knowledge *somewhere*, then they must coincide, while the second version states that, if the agents' *actual* posterior beliefs are common knowledge, then they must coincide.

Agreeing to disagree can be graphically illustrated for the case of two agents. In Figure 6, the set of all possible worlds Ω is partitioned horizontally in equivalence classes of worlds that yield a same posterior belief for agent *Alice* in some fixed event E. Similarly, the vertical slices represent equivalence classes with respect to worlds that induce a same posterior belief for *Bob* in E. Observe that the partition formed by the horizontal slices is coarser than *Alice*'s possibility partition, since Bayesian updating ensures that *Alice*'s posteriors remain constant throughout any cell of her possibility partition. Similarly, the partition formed by the vertical slices is coarser than *Bob*'s possibility partition. Moreover, the intersection of the horizontal and vertical slices forms a refined partition whose cells represent equivalence classes of worlds that induce a same posterior belief profile, i.e. the posterior of each agent remains constant throughout the cell. Given an auxiliary world $\hat{\omega}$ – a world which merely serves the supply of posterior beliefs that can potentially be generated given the constraints of the formal structure – the cell of this refined partition containing $\hat{\omega}$ represents the event $E' = \bigcap_{i \in I} \{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}$ and includes the event $CK(E')$. In particular, note that $CK(E')$ can be empty.

The claim of Aumann's agreement theorem can now be understood graphically: given a world $\hat{\omega}$, if the corresponding cell E' of the refined partition includes a non-empty $CK(E')$, then identical values for the agents' posterior beliefs obtain at $\hat{\omega}$, and hence also at all worlds throughout cell E'. Conversely, common knowledge of the agents' posteriors equals the empty set in all cells of the refined partition in which the posteriors of *Alice* and *Bob* differ. In particular, the equality of the two agents' posteriors throughout $CK(E')$ is established. Finally, note that in the specific

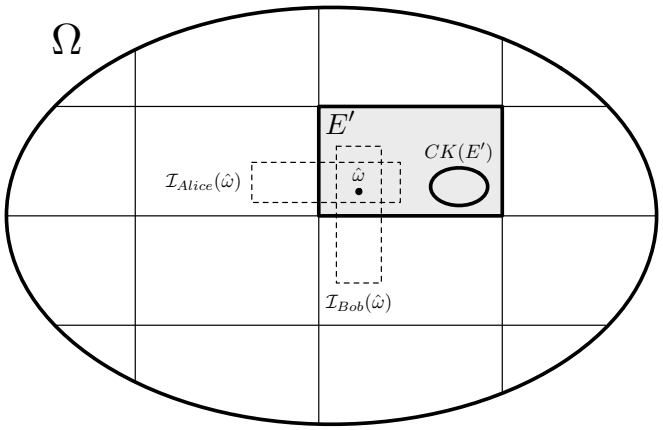


Figure 6 – Illustration of agreeing to disagree for the case of two agents.

Aumann structure represented in Figure 6, at world $\hat{\omega}$, none of the two agents know the true fact that their posteriors coincide, since each agent, respectively, considers possible a world at which the other agent's posterior belief is different. In contrast, for any world $\omega \in CK(E')$, the agents' posterior beliefs are not merely equal at such a world, but the agents also know that they coincide, know that they know that they coincide, etc.

5.3 Limit-Agreeing to Disagree

The question whether agents with a common prior belief can agree to disagree on their posterior beliefs is addressed from a topological point of view. The original hypotheses of Aumann's result are modified in that the epistemic operator common knowledge is replaced by the epistemic-topological operator limit knowledge. It is now shown that agents can indeed limit-agree to disagree.

Theorem 21. *There exist an Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$ equipped with a topology $\mathcal{T}$ on the event space $\mathcal{P}(\Omega)$, an event $E \subseteq \Omega$, and worlds $\omega, \hat{\omega} \in \Omega$ such that $\omega \in LK(\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\})$, as well as both $p(E \mid \mathcal{I}_i(\hat{\omega})) \neq p(E \mid \mathcal{I}_j(\hat{\omega}))$ and $p(E \mid \mathcal{I}_i(\omega)) \neq p(E \mid \mathcal{I}_j(\omega))$ for some agents $i, j \in I$.*

Proof. Consider the Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$, which is illustrated in Figure 7 and where $\Omega = \{\omega_n : n \geq 0\}$, $I = \{Alice, Bob\}$, $\mathcal{I}_{Alice} = \{\{\omega_{2n}, \omega_{2n+1}\} : n \geq 0\}$, $\mathcal{I}_{Bob} = \{\{\omega_0\}\} \cup \{\{\omega_{2n+1}, \omega_{2n+2}\} : n \geq 0\}$, as well as $p : \Omega \to \mathbb{R}$ is given by $p(\omega_n) = \frac{1}{2^{n+1}}$ for all $n \geq 0$. Note that the common prior belief function p is well defined since $\sum_{n\geq 0} \frac{1}{2^{n+1}} = 1$. Now, consider the event $E = \{\omega_{2n} : n \geq 1\}$, and the world $\omega_2 \in \Omega$. Besides, for sake of notational convenience, let the event $\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\omega_2))\}$ be denoted by E'. First of all, observe that $p(E \mid \mathcal{I}_{Alice}(\omega_2)) = \frac{2}{3}$ and $p(E \mid \mathcal{I}_{Bob}(\omega_2)) = \frac{1}{3}$. Moreover, $\{\omega' \in \Omega : p(E \mid \mathcal{I}_{Alice}(\omega')) = p(E \mid \mathcal{I}_{Alice}(\omega_2)) = \frac{2}{3}\} = \Omega \setminus \{\omega_0, \omega_1\}$ and $\{\omega' \in \Omega : p(E \mid \mathcal{I}_{Bob}(\omega')) = p(E \mid \mathcal{I}_{Bob}(\omega_2)) = \frac{1}{3}\} = \Omega \setminus \{\omega_0\}$, thus $E' = (\Omega \setminus \{\omega_0, \omega_1\}) \cap (\Omega \setminus \{\omega_0\}) = \Omega \setminus \{\omega_0, \omega_1\}$. Furthermore, the definitions of the possibility partitions of

Alice and *Bob* ensure that $K^m(E') = K^m(\Omega \setminus \{\omega_0, \omega_1\}) = \Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_{m+1}\}$, for all $m > 0$. Consequently, the sequence $(K^m(E'))_{m>0}$ is strictly shrinking and $CK(E') = \{\omega \in \Omega : (\bigwedge_{i \in I} \mathcal{I}_i)(\omega) \subseteq E'\} = \emptyset$. Now, consider the topology $\mathcal{T}$ on $\mathcal{P}(\Omega)$ defined by $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : \{\omega_0, \omega_1, \omega_2\} \notin O\} \cup \{\mathcal{P}(\Omega)\}$. Then, the only open neighbourhood of the event $\{\omega_0, \omega_1, \omega_2\}$ is $\mathcal{P}(\Omega)$, and all terms of the sequence $(K^m(E'))_{m>0}$ are contained in $\mathcal{P}(\Omega)$. Thus $(K^m(E'))_{m>0}$ converges to $\{\omega_0, \omega_1, \omega_2\}$. Moreover, for every event $F \in \mathcal{P}(\Omega)$ such that $F \neq \{\omega_0, \omega_1, \omega_2\}$, the singleton $\{F\}$ is open, and since $K^{m+1}(E') \subsetneq K^m(E')$ for all $m > 0$, the strictly shrinking sequence $(K^m(E'))_{m>0}$ will never remain in the open neighbourhood $\{F\}$ of F from some index onwards. Hence $(K^m(E'))_{m>0}$ does not converge to any such event F. Therefore the limit point $\{\omega_0, \omega_1, \omega_2\}$ of the strictly shrinking sequence $(K^m(E'))_{m>0}$ is unique, and $LK(E') = \lim_{m\to\infty} K^m(E') = \{\omega_0, \omega_1, \omega_2\}$. The event E' and its limit point $LK(E')$ are also illustrated in Figure 7. Next, consider the world ω_1. Note that $\omega_1 \in LK(E')$. Also, observe that $p(E \mid \mathcal{I}_{Alice}(\omega_2)) = \frac{2}{3} \neq \frac{1}{3} = p(E \mid \mathcal{I}_{Bob}(\omega_2))$ as well as $p(E \mid \mathcal{I}_{Alice}(\omega_1)) = 0 \neq \frac{1}{3} = p(E \mid \mathcal{I}_{Bob}(\omega_1))$. Finally, taking $\omega = \omega_1$ and $\hat{\omega} = \omega_2$ concludes the proof. □

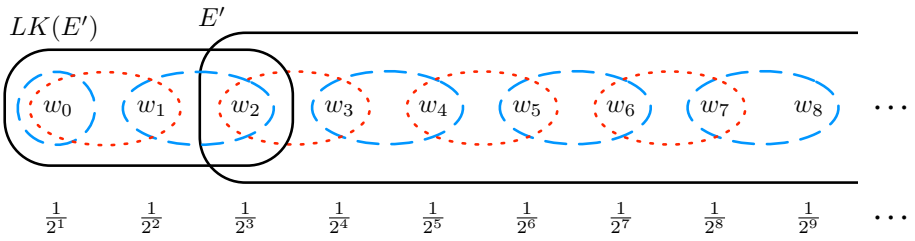


Figure 7 – Illustration of the Aumann structure described in the proof of Theorem 21. The dotted and dashed sets represent the possibility partitions of *Alice* and *Bob*, respectively. The fractions correspond to the prior probabilities associated to the possible worlds.

The preceding theorem counters Aumann's impossibility result in the sense of showing that agents can limit-agree to disagree. More precisely, agents may hold distinct actual posterior beliefs, while at the same time satisfying limit knowledge of their posteriors being equal to the specific values induced by a given auxiliary world. Hence, agents may agree in the sense of satisfying limit knowledge of their posteriors, while at the same time disagree in the sense of actually having different posterior beliefs.

The interactive situation depicted in the proof of Theorem 21 shows that agreeing to disagree becomes possible when common knowledge is substituted by limit knowledge. Thus, Aumann's impossibility result no longer holds when moving to a topologically enriched context. In such an epistemic-topological framework, agents can now be seen to have cognitive access to a further dimension in their reasoning that permits them to agree to disagree on their posterior beliefs. More precisely, the agents are in a limit situation of having higher-order mutual knowledge of their posteriors, which, in connection with the particular notion of closeness provided by the topology, leads them to actually possess different posterior beliefs.

Note that the epistemic model constructed in the proof of Theorem 21 could

actually be strengthened in the sense of generating different posterior beliefs for the agents not only at some but at all worlds contained in limit knowledge. For instance, taking as topology on the event space $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : \{\omega_n : n > 0\} \notin O\} \cup \{\mathcal{P}(\Omega)\}$ yields $LK(\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_{i\in I}(\hat{\omega}))\}) = \Omega \setminus \{\omega_0\}$, and hence different posterior beliefs for *Alice* and *Bob* at all worlds in $\Omega \setminus \{\omega_0\}$. More generally, it would actually be possible to make limit knowledge correspond to any event F by equipping the event space with the corresponding excluded point topology $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : F \notin O\} \cup \{\mathcal{P}(\Omega)\}$. This topology does not have any intuitive interpretation, but has merely be chosen as to keep the proof of Theorem 21 as simple as possible. However, in Section 5.4, a topological structure on the event space is given, which is based on intuitive properties and under which limit-agreeing to disagree also obtains.

The proof of Theorem 21 illustrates an epistemic situation in which limit knowledge of the agents' posterior beliefs holds concurrently with distinct actual posterior beliefs of the agents, in the particular case of the agents' posterior beliefs being nowhere common knowledge in the structure. Hence, the impression might arise that the relevance of limit knowledge for a possible disagreement only occurs in epistemic models in which common knowledge of the agents' posterior beliefs does not hold anywhere at all. However, it can be shown that the possibility of such a disagreement may also emerge in situations in which common knowledge of the agents' posteriors actually holds somewhere in the structure. Hence, disagreement induced by limit knowledge does not depend on the existence or non-existence of common knowledge of the agents' posteriors beliefs in the epistemic model. From an interpretative point of view, limit knowledge of the agents' posteriors might be consistent with a disagreement between the agents irrespective of whether these posteriors would have been publicly disclosed somewhere, or not. Also, the proof of Theorem 21 describes an interactive situation in which limit knowledge of distinct agents' posterior beliefs obtains simultaneously with differing actual posterior beliefs of the agents. Thus, the impression might arise that a possible disagreement could only be elicited by limit knowledge of already distinct posteriors. However, it can be shown that agents may actually disagree, while having limit knowledge of identical posterior beliefs. Countering the preceding two possible impressions, the following example indeed depicts an epistemic situation in which a disagreement on the agents' actual posterior beliefs is induced by limit knowledge of identical posterior beliefs, while common knowledge of these posteriors also holds somewhere in the structure.

Example 22. Consider the Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$, where $\Omega = \{\omega_n : n \geq 0\}$, $I = \{Alice, Bob\}$, $I_{Alice} = \{\{\omega_0\}, \{\omega_1\}\} \cup \{\{\omega_{2n}, \omega_{2n+1}\} : n > 0\}$, $I_{Bob} = \{\{\omega_0\}\} \cup \{\{\omega_{2n+1}, \omega_{2n+2}\} : n \geq 0\}$, and $p : \Omega \to \mathbb{R}$ is given by $p(\omega_n) = \frac{1}{2^{n+1}}$ for all $n \geq 0$. This structure is illustrated in Figure 8. Note that the common prior belief function p is well defined since $\sum_{n\geq 0} \frac{1}{2^{n+1}} = 1$. Now, consider the event $E = \{\omega_1\}$ and the world ω_0. Besides, for sake of notational convenience, let the event $\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\omega_0))\}$ be denoted by E'. First of all, observe that $p(E \mid \mathcal{I}_{Alice}(\omega_0)) = 0$ as well as $p(E \mid \mathcal{I}_{Bob}(\omega_0)) = 0$. Moreover, $\{\omega' \in \Omega : p(E \mid \mathcal{I}_{Alice}(\omega')) = p(E \mid \mathcal{I}_{Alice}(\omega_0)) = 0\} = \Omega \setminus \{\omega_1\}$ and $\{\omega' \in \Omega : p(E \mid \mathcal{I}_{Bob}(\omega')) = p(E \mid \mathcal{I}_{Bob}(\omega_0)) = 0\} = \Omega \setminus \{\omega_1, \omega_2\}$, whence $E' = (\Omega \setminus \{\omega_1\}) \cap (\Omega \setminus \{\omega_1, \omega_2\}) = \Omega \setminus \{\omega_1, \omega_2\}$. Fur-

thermore, the definitions of the possibility partitions of *Alice* and *Bob* ensure that $K^m(E') = K^m(\Omega \setminus \{\omega_1, \omega_2\}) = \Omega \setminus \{\omega_1, \omega_2, \ldots, \omega_{m+2}\}$, for all $m > 0$. Consequently, the sequence $(K^m(E'))_{m>0}$ is strictly shrinking. Now, consider the topology $\mathcal{T}$ on $\mathcal{P}(\Omega)$ defined by $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : \{\omega_1, \omega_2\} \notin O\} \cup \{\mathcal{P}(\Omega)\}$. Then, the only open neighbourhood of the event $\{\omega_1, \omega_2\}$ is $\mathcal{P}(\Omega)$, and all terms of the sequence $(K^m(E'))_{m>0}$ are contained in $\mathcal{P}(\Omega)$. Thus $(K^m(E'))_{m>0}$ converges to $\{\omega_1, \omega_2\}$. Moreover, for every event $F \in \mathcal{P}(\Omega)$ such that $F \neq \{\omega_1, \omega_2\}$, the singleton $\{F\}$ is open, and since $K^{m+1}(E') \subsetneq K^m(E')$ for all $m > 0$, the strictly shrinking sequence $(K^m(E'))_{m>0}$ will never remain in the open neighbourhood $\{F\}$ of F from some index onwards. Hence $(K^m(E'))_{m>0}$ does not converge to any such event F. Therefore the limit point $\{\omega_1, \omega_2\}$ of the strictly shrinking sequence $(K^m(E'))_{m>0}$ is unique, and $LK(E') = \lim_{m\to\infty} K^m(E') = \{\omega_1, \omega_2\}$. Besides, note that $CK(E') = \{\omega \in \Omega : (\bigwedge_{i\in I} \mathcal{I}_i)(\omega) \subseteq E'\} = \{\omega_0\}$. Next, consider the world ω_1. Observe that $\omega_1 \in LK(E')$ and $p(E \mid \mathcal{I}_{Alice}(\omega_1)) = 1 \neq \frac{2}{3} = p(E \mid \mathcal{I}_{Bob}(\omega_1))$. Thus, $CK(E') \neq \emptyset$, and world ω_1 satisfies both conditions $\omega_1 \in LK(E')$ as well as $p(E \mid \mathcal{I}_{Alice}(\omega_1)) \neq p(E \mid \mathcal{I}_{Bob}(\omega_1))$. ♣

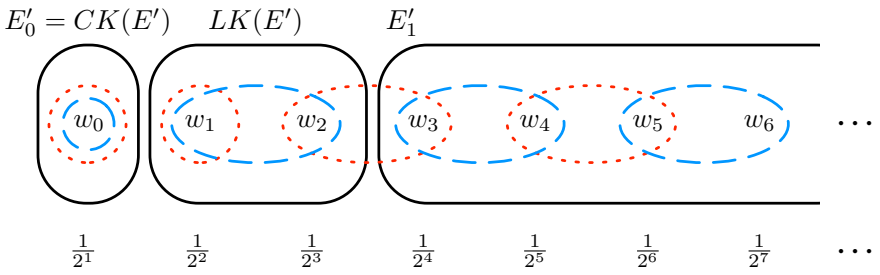


Figure 8 – Illustration of the Aumann structure used in Example 22. The dotted and dashed sets represent the possibility partitions of *Alice* and *Bob*, respectively. The fractions correspond to the prior probabilities associated with the possible worlds. Note that $E' = E'_0 \cup E'_1$.

Observe that the structure of the two agents' possibility partitions in the proof of Theorem 21 and Example 22 is similar to the structure of the partitions in RUBINSTEIN (1989)'s electronic mail game. Indeed, the resemblance lies in the existence of an infinite chain-type pattern which consecutively links information cells of the two agents by a single world in the intersection of the respective two cells, and where each information cell only contains two worlds. In our framework, such a pattern ensures that the sequence of iterated mutual knowledge is strictly shrinking, which is a necessary condition for limit knowledge to differ from common knowledge.

More generally, observe that all possible ways of limit-agreeing to disagree can actually be classified into three mutually exclusive cases.

First of all, disagreement on the agents' actual posteriors may be induced by limit knowledge of already distinct posteriors, while common knowledge of these posteriors is empty. Such agreeing to disagree is illustrated by the interactive situation depicted in the proof of Theorem 21.

Secondly, disagreement on the agents' actual posteriors can be induced by limit knowledge of identical posteriors, while common knowledge of these posteriors is

non-empty. Such agreeing to disagree is illustrated in Example 22.

Thirdly, disagreement on the agents' actual posteriors can also be induced by limit knowledge of identical agents' posteriors, while common knowledge of these posteriors is empty. To see this, consider the Aumann structure given in the proof of Theorem 21, the event $E = \{\omega_0\}$, the world ω_2, and the topology on the event space $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : \{\omega_0, \omega_1\} \notin O\} \cup \{\mathcal{P}(\Omega)\}$. It thus follows that $p(E \mid \mathcal{I}_{Alice}(\omega_2)) = p(E \mid \mathcal{I}_{Bob}(\omega_2)) = 0$ and $E' = \bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\omega_2))\} = \Omega \setminus \{\omega_0, \omega_1\}$. Then, $CK(E') = \emptyset$ and $LK(E') = \{\omega_0, \omega_1\}$, as well as both $\omega_1 \in LK(E')$ and $p(E \mid \mathcal{I}_{Alice}(\omega_1)) = \frac{2}{3} \neq 0 = p(E \mid \mathcal{I}_{Bob}(\omega_1))$.

The fourth possibility of a disagreement on the agents' actual posteriors based on limit knowledge of already distinct posteriors and with non-emptiness of common knowledge of these posteriors is excluded by Aumann's agreement theorem.

Besides, in the epistemic-topological situations described in the proof of Theorem 21 as well as in Example 22, limit knowledge is not factive, i.e., the relation $LK(E') \subseteq E'$ does not hold for the considered event E'. However, the possibility to limit-agree to disagree established in Theorem 21 does not directly follow from the non-factiveness of limit knowledge. We now show that agents can limit-agree to disagree with factive limit knowledge, and that in this case, the distinct actual posteriors are induced by limit knowledge of already distinct posteriors as well as common knowledge of these posteriors being empty.

Lemma 23. *There exist an Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$ equipped with a topology $\mathcal{T}$ on the event space $\mathcal{P}(\Omega)$, an event $E \subseteq \Omega$, and worlds $\omega, \hat{\omega} \in \Omega$ such that $LK(\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}) \subseteq \bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}$, $\omega \in LK(\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\})$, as well as $p(E \mid \mathcal{I}_i(\omega)) \neq p(E \mid \mathcal{I}_j(\omega))$ for some agents $i, j \in I$. In this case, $CK(\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}) = \emptyset$ and $p(E \mid \mathcal{I}_i(\hat{\omega})) \neq p(E \mid \mathcal{I}_j(\hat{\omega}))$ for some agents $i, j \in I$.*

Proof. First, the existence of an Aumann structure and a topology satisfying the postulated properties is established. Consider the Aumann structure $\mathcal{A}$ and the events E and E' given in the proof of Theorem 21. Let the topology $\mathcal{T}$ on the event space $\mathcal{P}(\Omega)$ be defined by $\mathcal{T} = \{O \subseteq \mathcal{P}(\Omega) : \{\omega_2\} \notin O\} \cup \{\mathcal{P}(\Omega)\}$. It follows that $LK(E') = \lim_{m\to\infty} K^m(E') = \{\omega_2\} \subseteq \Omega \setminus \{\omega_0, \omega_1\} = E'$, and $p(E \mid \mathcal{I}_{Alice}(\omega_2)) = \frac{2}{3} \neq \frac{1}{3} = p(E \mid \mathcal{I}_{Bob}(\omega_2))$. Taking $\omega = \hat{\omega} = \omega_2$ concludes the first part of the proof. Next, it is shown that if an Aumann structure and a topology satisfy the postulated properties, then the corresponding posteriors are distinct and common knowledge of these posteriors is thus empty. Consider some Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$, some topology $\mathcal{T}$ on the event space $\mathcal{P}(\Omega)$, some event E, and some worlds $\omega, \hat{\omega} \in \Omega$ satisfying the postulated conditions. Let the event $\bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\hat{\omega}))\}$ be denoted by E'. Since $\omega \in LK(E')$ and $LK(E') \subseteq E'$ both hold by the postulated conditions, it follows that $\omega \in E'$, i.e. $p(E \mid \mathcal{I}_i(\omega)) = p(E \mid \mathcal{I}_i(\hat{\omega}))$, for all $i \in I$. Moreover, as the postulated conditions ensure that $p(E \mid \mathcal{I}_i(\omega)) \neq p(E \mid \mathcal{I}_j(\omega))$ for some agents $i, j \in I$, it is the case that $p(E \mid \mathcal{I}_i(\hat{\omega})) \neq p(E \mid \mathcal{I}_j(\hat{\omega}))$ also obtains for some agents $i, j \in I$. Now, the contraposition of Aumann's Agreement Theorem (Version 1) directly implies that $CK(E') = \emptyset$. □

Finally, agreeing to disagree with limit knowledge can be graphically illustrated for the case of two agents. A particular interactive situation is represented in Figure 9. Again, as in Figure 6, the event space is partitioned in equivalence classes of worlds inducing a same posterior belief profile in some underlying event E. Hence, the event E' denotes the equivalence class of worlds inducing the same posterior beliefs for all agents as at the auxiliary world $\hat{\omega}$. In the considered interactive situation, the event $CK(E')$ is non-empty, and the topology on the event space implies that the event $LK(E')$ is well-defined, distinct from $CK(E')$, and not even included in the event E' itself. Since $CK(E')$ is non-empty, Aumann's agreement theorem ensures that the agents' posterior beliefs induced by the auxiliary world $\hat{\omega}$ coincide, and thus any world in the equivalence class E' also induces identical posterior beliefs for all agents. Therefore, the posterior beliefs that are both common knowledge as well as limit knowledge are identical for all agents. Moreover, note that, as ω_1 lies in the equivalence class E', the agents' posterior beliefs at world ω_1 are the same as the ones induced by the auxiliary world $\hat{\omega}$, and thus all identical. Consequently, since ω_1 is also contained in both $CK(E')$ as well as in $LK(E')$, agents do agree on identical posteriors at ω_1, both with common knowledge and with limit knowledge. Besides, the position of the world ω_2 relative to ω_1 in Figure 9 ensures that *Alice* and *Bob* hold distinct posterior beliefs at ω_2. Similarly, the agents have different posterior beliefs at ω_3. Since ω_2 and ω_3 are contained in $LK(E')$, agents actually agree to disagree with limit knowledge at both worlds. Observe that, although being based on limit knowledge of the same posterior beliefs, the disagreements at ω_2 and ω_3 differ. Furthermore, both such disagreements are in fact induced by limit knowledge of equal posteriors, as the posteriors of *Alice* and *Bob* coincide throughout E'.

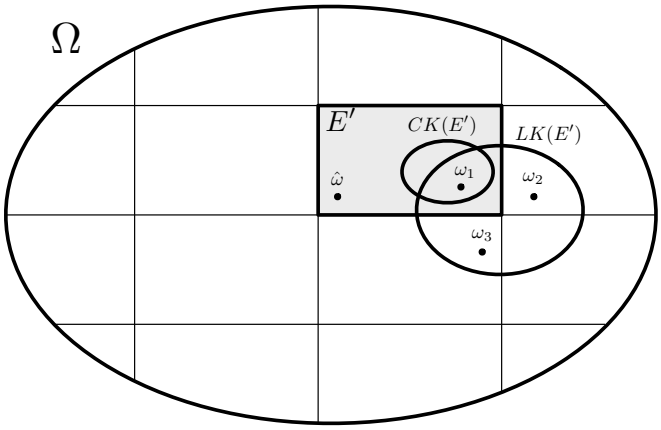


Figure 9 – Illustration of limit-agreeing to disagree for the case of two agents.

5.4 A Representative Example

The extension of the standard set-based approach to interactive epistemology with a topological dimension has been shown to enable the possibility for agents to limit-

agree to disagree on their posterior beliefs. The question then arises whether limit-agreeing to disagree remains possible in interactive situations, where the topologies are based on intuitive properties. A topology describing a specific agents' perception of the event space is now presented. Accordingly, lower iterated mutual knowledge up to some finite level is grasped by the agents in a more refined manner than higher iterations from that level onwards. Such a property seems natural, since real world agents typically only have distinguished cognitive access to iterated knowledge claims up to some finite level, in contrast to the idealized agents that can equally well conceive of any layer of the complete hierarchy of interactive knowledge. For this intuitive topology it is then shown that agreeing to disagree with limit knowledge is possible. Besides, this intuitive topology establishes that limit knowledge is identical to iterated mutual knowledge for some finite level m, i.e. it is equal to almost common knowledge, a concept due to RUBINSTEIN (1989). The example could thus also be seen as a contribution to the literature of bounded reasoning. However, it differs from models of k-level reasoning, which express a specific and different kind of finite level reasoning in the particular context of games.[1]

Towards this purpose, consider an Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$ and an event E. Furthermore, for any world $\omega \in \Omega$, let E'_ω denote the event consisting of all worlds that induce the same posterior beliefs in E for all agents as at ω, i.e. $E'_\omega = \bigcap_{i\in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\omega))\}$. Note that constancy of the agents' posterior beliefs in E yields an equivalence relation on the set of possible worlds, and hence every E'_ω represents an equivalence class of worlds. Consequently, the collection $\mathcal{C} = \{E'_\omega : \omega \in \Omega\}$ of all equivalence classes of worlds that induce a same posterior belief profile forms a partition of Ω. Given some event E and some index $m^* > 0$, the epistemically-based topology $\mathcal{T}_{E,m^*}$ is defined as the topology on the event space $\mathcal{P}(\Omega)$ generated by the subbase

$$
\begin{aligned}
& \{\{K^m(E'_\omega) : m \geq 0\} : \omega \in \Omega\} \\
\cup \quad & \{\mathcal{P}(\Omega) \setminus \{K^m(E'_\omega) : m \geq 0 \text{ and } \omega \in \Omega\}\} \\
\cup \quad & \{\{K^m(E'_\omega)\} : 0 \leq m < m^* \text{ and } \omega \in \Omega\} \\
\cup \quad & \{\{K^{m^*+j}(E'_\omega) : 0 < j \leq n\} : n > 0 \text{ and } \omega \in \Omega\}.
\end{aligned}
$$

The topology $\mathcal{T}_{E,m^*}$ is illustrated in Figure 10, where the sequence $(K^m(E'_\omega))_{m\geq 0}$ is represented by a horizontal sequence of points for each $\omega \in \Omega$, and open sets of the subbase by circle-type shapes around these points. In this topology, the closeness relation between events is represented by means of the T_0 and T_2 separation properties.[2]

[1] Loosely speaking, k-level reasoning restricts a belief hierarchy in the particular context of games such that random play is assumed at the first level, and only best responses to the respective preceding levels are admitted at the iterated levels up to some finite level k. In fact, CRAWFORD, COSTA-GOMES, and IRIBERRI (2013) provide a recent overview on the literature of k-level reasoning and other theories of finite reasoning.

[2] Given a topological space $(A, \mathcal{T})$, two points in A are called *T_2-separable*, if there exist two disjoint $\mathcal{T}$-open neighbourhoods of these two points. Moreover, two points in A are called *T_0-separable*, if there exists a $\mathcal{T}$-open set containing precisely one of these two points. Note that T_2-separability implies T_0-separability. In fact, two events that are T_0-separable but not T_2-separable can be said to be closer to each other than two events that are both T_0-separable as well as T_2-separable.

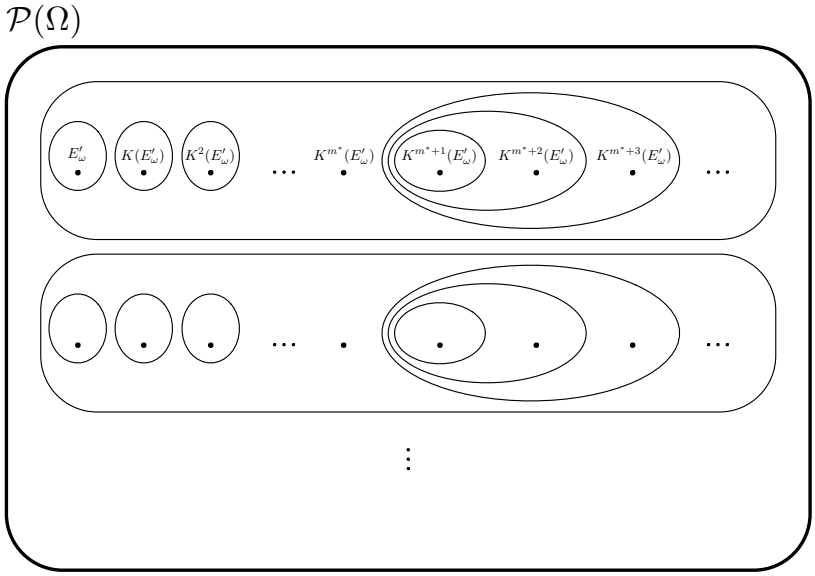


Figure 10 – Illustration of the topology $\mathcal{T}_{E,m^*}$.

The topology $\mathcal{T}_{E,m^*}$ reveals a specific agent perception of the event space, according to which the agents express a more refined distinction between the m^* first iterated mutual knowledge of their posterior beliefs in E than between the remaining ones. This specific perception is formally reflected by two separation properties satisfied by the topology $\mathcal{T}_{E,m^*}$.

Firstly, given two events X and Y, if X and Y are two distinct terms of the same sequence $(K^m(E'_\omega))_{m>0}$, such that $X = K^{m_1}(E'_\omega)$ and $Y = K^{m_2}(E'_\omega)$, for some $\omega \in \Omega$ and $m_1, m_2 < m^*$, then X and Y are T_2-separable, and therefore also T_0-separable. Secondly, if X and Y are two different elements of the same sequence $(K^m(E'_\omega))_{m>0}$, such that $X = K^{m_1}(E'_\omega)$ and $Y = K^{m_2}(E'_\omega)$, for some $\omega \in \Omega$ and $m_1, m_2 > m^*$, then X and Y are T_0-separable, yet not T_2-separable. According to these two separation properties, agents have access to a more refined distinction between the m^* first iterated knowledge claims of their posterior beliefs in E than between the iterated mutual knowledge claims of order strictly larger than m^*. In other words, iterated mutual knowledge claims are only precisely discerned up to a given amount of iterations, and thereafter the higher iterations become less distinguishable for the agents. Also, from a bounded rationality point of view, the agents' perception of higher-order mutual knowledge due to the topology $\mathcal{T}_{E,m^*}$ reflects that people typically lose track from some iteration level onwards when reasoning about higher-order mutual knowledge.

Furthermore, the topology $\mathcal{T}_{E,m^*}$ notably satisfies the following epistemic-topological property: for any event E'_ω, if the sequence $(K^m(E'_\omega))_{m>0}$ is strictly shrinking, then $LK(E'_\omega) = K^{m^*}(E'_\omega)$. Indeed, suppose that the sequence $(K^m(E'_\omega))_{m>0}$ is strictly shrinking. Then, by definition of $\mathcal{T}_{E,m^*}$, the only open neighbourhoods of $K^{m^*}(E'_\omega)$ are $\mathcal{P}(\Omega)$ and $\{K^m(E'_\omega) : m \geq 0\}$. Since both sets contain all terms of the sequence $(K^m(E'_\omega))_{m>0}$, it follows that $K^{m^*}(E'_\omega)$ is a limit point of the sequence

$(K^m(E'_\omega))_{m>0}$.

Moreover, this limit point is actually unique. To see this, consider $F \in \mathcal{P}(\Omega)$ such that $F \neq K^{m^*}(E'_\omega)$. Then either $F = K^m(E'_{\omega'})$ for some $m < m^*$ and some $\omega' \in \Omega$, or $F = K^m(E'_{\omega'})$ for some $m > m^*$ and some $\omega' \in \Omega$, or $F = K^{m^*}(E'_{\omega'})$ for some $\omega' \neq \omega$, or $F \neq K^m(E'_{\omega'})$ for all $m \geq 0$ and all $\omega' \in \Omega$. These four mutually exclusive cases are now considered in turn. First of all, if $F = K^m(E'_{\omega'})$ for some $m < m^*$ and some $\omega' \in \Omega$, then $\{K^m(E'_{\omega'})\}$ is an open neighbourhood of F. Since the sequence $(K^m(E'_\omega))_{m>0}$ is strictly shrinking, it can then not be the case that the singleton open neighbourhood $\{K^m(E'_{\omega'})\}$ of F contains all terms of the sequence $(K^m(E'_\omega))_{m>0}$ from some index onwards. Therefore F is not a limit point of the sequence $(K^m(E'_\omega))_{m>0}$. Secondly, if $F = K^m(E'_{\omega'})$ for some $m > m^*$ and some $\omega' \in \Omega$, then $\{K^{m^*+j}(E'_{\omega'}) : 0 < j \leq m - m^*\}$ is an open neighbourhood of F. Since the set $\{K^{m^*+j}(E'_{\omega'}) : 0 < j \leq m - m^*\}$ is finite, F cannot be a limit point of the sequence $(K^m(E'_\omega))_{m>0}$. Thirdly, if $F = K^{m^*}(E'_{\omega'})$ for some $\omega' \neq \omega$, then $\{K^n(E'_{\omega'}) : n \geq 0\}$ is an open neighbourhood of F. Moreover, since $K^{m^*}(E'_\omega) \neq K^{m^*}(E'_{\omega'}) = F$, it directly follows that $E'_\omega \neq E'_{\omega'}$. Yet since $\mathcal{C} = \{E'_{\omega''} : \omega'' \in \Omega\}$ is a partition of Ω, it holds that $E'_\omega \cap E'_{\omega'} = \emptyset$. Moreover, as $K^m(E'_\omega) \subseteq E'_\omega$ for all $m \geq 0$, and $K^n(E'_{\omega'}) \subseteq E'_{\omega'}$ for all $n \geq 0$, as well as $E'_\omega \cap E'_{\omega'} = \emptyset$, it follows that $K^m(E'_\omega) \neq K^n(E'_{\omega'})$ for all $m, n \geq 0$. Thus the open neighbourhood $\{K^n(E'_{\omega'}) : n \geq 0\}$ of F contains no term of the sequence $(K^m(E'_\omega))_{m>0}$ whatsoever. Therefore, F is not a limit point of the sequence $(K^m(E'_\omega))_{m>0}$. Fourthly, if $F \neq K^m(E'_{\omega'})$ for all $m \geq 0$ and all $\omega' \in \Omega$, then $\mathcal{P}(\Omega) \setminus \{K^m(E'_\omega) : m \geq 0 \text{ and } \omega \in \Omega\}$ is an open neighbourhood of F. Yet this set contains no term of the sequence $(K^m(E'_\omega))_{m>0}$. Thus F is not a limit point of the sequence $(K^m(E'_\omega))_{m>0}$. To summarize, there consequently exists no $F \neq K^{m^*}(E'_\omega)$ which is a limit point of the sequence $(K^m(E'_\omega))_{m>0}$. Hence, the limit point $K^{m^*}(E'_\omega)$ of the sequence $(K^m(E'_\omega))_{m>0}$ is unique, and therefore, $LK(E'_\omega) = \lim_{m\to\infty} K^m(E'_\omega) = K^{m^*}(E'_\omega)$.

Furthermore, since the sequence $(K^m(E'_\omega))_{m>0}$ is strictly shrinking, $CK(E'_\omega) = \bigcap_{m>0} K^m(E'_\omega) \subsetneq K^{m^*}(E'_\omega)$, and hence $CK(E'_\omega) \neq LK(E'_\omega)$.

Note that if the event space is equipped with the topology $\mathcal{T}_{E,m^*}$, the epistemic-topological event $LK(E')$ actually coincides with RUBINSTEIN (1989)'s notion of almost common knowledge if m^* is large, i.e. $LK(E') = K^{m^*}(E')$. Thus, agents with topological mental states according to $\mathcal{T}_{E,m^*}$ who have limit knowledge actually reason in line with almost common knowledge. More precisely, if agents can only accurately conceive of higher-order interactive knowledge up to some fixed level, then they reason with limit knowledge if and only if they reason with almost common knowledge up to that level only. In fact, the connection between *cognitive-topologically* perceiving iterated knowledge distinctly only up to some finite level and *epistemically* reasoning in line with almost common knowledge up to that level does seem natural. Thus, the topology $\mathcal{T}_{E,m^*}$ provides an intuitive topological foundation for RUBINSTEIN (1989)'s almost common knowledge in terms of the agents' cognitive perception of the event space.

Finally, the following example describes an interactive situation, in which the intutive topology $\mathcal{T}_{E,m^*}$ provides a possibility for the agents to agree to disagree on their posterior beliefs with limit knowledge.

Example 24. Consider the Aumann structure $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i\in I}, p)$, where $\Omega = \{\omega_n :$

$n \geq 0\}$, $I = \{Alice, Bob\}$, $I_{Alice} = \{\{\omega_0\}, \{\omega_1, \omega_2\}, \{\omega_3, \omega_4, \omega_5, \omega_6\}, \{\omega_7, \omega_8, \omega_9\}\} \cup \{\{\omega_{2n}, \omega_{2n+1}\} : n \geq 5\}$, $I_{Bob} = \{\{\omega_0, \omega_1, \omega_2, \omega_3, \omega_4\}, \{\omega_5, \omega_6, \omega_7, \omega_8\}\} \cup \{\{\omega_{2n+1}, \omega_{2n+2}\} : n \geq 4\}$, and $p : \Omega \to \mathbb{R}$ is given by $p(\omega_n) = \frac{1}{2^{n+1}}$ for all $n \geq 0$. Also, consider the event $E = \{\omega_1, \omega_5\} \cup \{\omega_{2n} : n \geq 1\}$ and the world ω_{10}. Besides, for sake of notational convenience, let the event $\bigcap_{i \in I}\{\omega' \in \Omega : p(E \mid \mathcal{I}_i(\omega')) = p(E \mid \mathcal{I}_i(\omega_{10}))\}$ be denoted by E'. First of all, observe that the computation of the posterior beliefs of *Alice* and *Bob* gives a variety of distinct values for the first ten worlds $\{\omega_0, \omega_1, \ldots, \omega_9\}$, as well as $p(E \mid \mathcal{I}_{Alice}(\omega_n)) = \frac{2}{3}$ and $p(E \mid \mathcal{I}_{Bob}(\omega_n)) = \frac{1}{3}$, for all $n \geq 10$. It follows that $\{\omega' \in \Omega : p(E \mid \mathcal{I}_{Alice}(\omega')) = p(E \mid \mathcal{I}_{Alice}(\omega_{10}))\} = \Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_9\}$ and $\{\omega' \in \Omega : p(E \mid \mathcal{I}_{Bob}(\omega')) = p(E \mid \mathcal{I}_{Bob}(\omega_{10}))\} = \Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_8\}$, thus $E' = (\Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_9\}) \cap (\Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_8\}) = \Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_9\}$. Moreover, the definitions of the possibility partitions of *Alice* and *Bob* ensure that $K^m(E') = \Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_{m+9}\}$, for all $m > 0$. Consequently, the sequence $(K^m(E'))_{m>0}$ is strictly shrinking and $CK(E') = \bigcap_{m>0} K^m(E') = \emptyset$. Now, let $m^* > 0$ be some index and suppose that $\mathcal{P}(\Omega)$ is equipped with the topology $\mathcal{T}_{E,m^*}$. Since the sequence $(K^m(E'))_{m>0}$ is strictly shrinking, the definition of this topology ensures that $LK(E') = K^{m^*}(E') = \Omega \setminus \{\omega_0, \omega_1, \ldots, \omega_{m^*+9}\}$. Consequently, the computations of the posterior beliefs of *Alice* and *Bob* give $p(E \mid \mathcal{I}_{Alice}(\omega)) = \frac{2}{3}$ and $p(E \mid \mathcal{I}_{Bob}(\omega)) = \frac{1}{3}$, for all $\omega \in LK(E')$. In other words, for all $\omega \in LK(E')$, it holds that $p(E \mid \mathcal{I}_{Alice}(\omega)) \neq p(E \mid \mathcal{I}_{Bob}(\omega))$. ♣

In the preceding example, a situation is provided where agents with an intuitive perception of the event space do agree to disagree. Yet, as limit knowledge coincides with RUBINSTEIN (1989)'s almost common knowledge, the example also shows that Aumann's agreement theorem is not robust in the sense that agents having almost common knowledge of posteriors can hold distinct posterior beliefs.

5.5 DISCUSSION

We showed that in a topologically extended epistemic model, agents can limit-agree to disagree. More precisely, if Bayesian agents have a common prior belief as well as limit knowledge of their posteriors beliefs, then their actual posterior beliefs may indeed differ. This possibility result contrasts with AUMANN (1976)'s impossibility to agree to disagree. Moreover, we showed that limit-agreeing to disagree is also possible in cases where the underlying topologies reveal cogent agent perception of the event space. Indeed, in our representative example, limit-agreeing to disagree is possible in an epistemic-topologcal situation where limit knowledge coincides with RUBINSTEIN (1989)'s almost common knowledge. Overall, these considerations illustrate the non-robustness of AUMANN (1976)'s agreement theorem to situations where common knowledge is replaced by limit knowledge of the posteriors.

However, note that it is impossible for agents to limit-agree to disagree in the case of finite Aumann structures as well as in the case of infinite Aumann structures in which the sequence of iterated mutual knowledge is not strictly shrinking. Indeed, as shown in Chapter 3, in such cases, the epistemic-topological operator limit knowledge necessarily coincides with the purely epistemic operator common knowledge, and consequently, Aumann's impossibility result does apply.

The possibility of agreeing to disagree with limit knowledge – as finitely iter-

ated mutual knowledge – in a topologically enriched epistemic structure can also be seen in the context of the growing literature on k-level reasoning and other theories on bounded reasoning. It would be intriguing for future work to investigate such models of finite thinking from an epistemic-topological point of view. An interesting recent point of departure could be KETS (2014) and KETS (2014)'s theory of finite depth reasoning. Accordingly, the language of game-playing agents is restricted such that they can only reason about higher-order beliefs up to some level k. This is related to our intuitive topology in Section 5.4, which also restricts agents' reasoning to finitely iterated mutual knowledge. Even though any higher-order mutual knowledge is – in contrast to Kets' models – part of the agents' language, they cannot conceive of it in a precise but only in a "blurred" way. Hence, a relevant question would be what topological conditions need to generally be invoked on standard epistemic structures for games, such that the players' reasoning remains restricted as in Kets' style type spaces.

For future work, the possibility for agents to agree to disagree with limit knowledge is expected to be further analyzed in the context of other epistemically-based as well as agent specific topologies, i.e., in all those topological contexts revealing some relevant or plausible notion of closeness between events. More generally, it would be of specific interest to provide a precise characterization of the topologies that enable or exclude the possibility to limit-agree to disagree. The classical impossibility to agree to disagree (with common knowledge) would thus appear as a particular case of this global topological characterization: the specific situations where limit knowledge coincide with common knowledge.

6 CONCLUSION

SUMMARY. The standard set-based approach to interactive epistemology lacks the possibility to model some formal notion of closeness between worlds, events, or other features expressible in the underlying semantic structures. Accordingly, we considered an epistemic-topological approach to interactive epistemology and epistemic game theory which enables the modelling of some notion of proximity between events.

In this context, we introduced the new epistemic-topological operator *limit knowledge*, defined as the topological limit of higher-order mutual knowledge claims, according to some topology on the event space (BACH and CABESSA (2009)). Limit knowledge can be understood as the event which is approached by the sequence of iterated mutual knowledge, with respect to the notion of closeness induced by the underlying topology. It can be viewed as some kind of generalized epistemic-topological concept which, for every possible underlying topology, becomes an operator with precise meaning. In general, limit knowledge differs from common knowledge as well as from other approximations of it, such as RUBINSTEIN (1989)'s almost common knowledge as well as MONDERER and SAMET (1989)'s common p-belief.

We showed that limit knowledge can yield to some relevant characterizations of solution concepts in games (BACH and CABESSA (2012), BACH and CABESSA (2009)). We provided an example of a Cournot-type game where the behavioral implications of limit knowledge of rationality strictly refine those of common knowledge of rationality. Moreover, we showed that limit knowledge of rationality is capable of characterizing any possible solution concept, under some appropriate epistemic-topological conditions.

Furthermore, we revisited AUMANN (1976)'s "no-agreeing to disagree theorem" in the epistemic-topological context of limit knwoledge (BACH and CABESSA (2011), BACH and CABESSA (2016)). We proved that the impossibility to agree to disagree does no longer hold when the epistemic hypothesis of common knowledge of the posteriors is replaced by that of limit knowledge of the posteriors. We also provided an epistemic-topological foundation for RUBINSTEIN (1989)'s notion of almost common knowledge. These results show that Aumann's agreement theorem is not robust when considered from a more general epistemic-topological perspective, where limit knowledge is substituted to common knowledge.

The epistemic-topological operator of "limit knowledge" represents a relevant alternative to that of "common knowledge", which has often been argued as being inappropriate or paradoxical for the modelling of a deeply shared knowledge,

see MORRIS (2002). It permits to model some reasoning patterns of game-theoretic agents based on a notion of proximity between their higher-order knowledge claims, rather than on purely logical considerations.

TOPOLOGICAL AUMANN STRUCTURES. Aumann structures represent an abstract framework in which, on the basis of epistemic assumptions, the reasoning of agents about events can be formalized. By enriching this epistemic framework with a topological dimension, one obtains an epistemic-topological semantics which permits to model richer agent perceptions of the event and state spaces, and subsequently, wider agent reasoning patterns that do not only depend on mere epistemic but also on topological features of the underlying interactive situations.

For instance, as already mentioned in Chapter 1, the event *It is cloudy in London* seems to be closer to the event *It is raining in London* than the event *It is sunny in London*. Now, agents may make identical decisions only being informed of the truth of some event within a class of *close* events. Indeed, *Alice* might decide to stay at home not only in the case of it raining outside, but also in the case of events perceived by her to be similar, such as it being cloudy outside.

Accordingly, we envision the consideration of a more general epistemic-topological framework – the *topological Aumann structures* – comprising topologies not only on the event space but also on the state space. Such an extension permits an explicit consideration of a notion of closeness between events and between worlds, and hence, allows to model common agent perceptions of the event and state spaces.[1] The definition of this epistemic-topological structure would be the following:

Definition 25. A *topological Aumann structure* is a tuple $\mathcal{A}_{\mathcal{T}} = (\mathcal{A}, \mathcal{T}^{\Omega}, \mathcal{T}^{\mathcal{P}(\Omega)})$, where $\mathcal{A} = (\Omega, (\mathcal{I}_i)_{i \in I}, p)$ is an Aumann structure, $\mathcal{T}^{\Omega}$ is a topology on the state space Ω, and $\mathcal{T}^{\mathcal{P}(\Omega)}$ is a topology on the event space $\mathcal{P}(\Omega)$.

In this context, it might be of distinguished interest to base topologies on first principles, such as epistemic axioms or natural closeness properties. In line with this perspective, the topology provided in Section 5.4 reflects the natural agent perception for which iterated mutual knowledge becomes imprecise from some level onwards.

Besides, in order to model subjective rather than common agent perceptions of the event and state spaces, one could also assign specific and potentially distinct topologies for every agent. A collective topology reflecting a common closeness perception could then be constructed on the basis of the particular agent topologies. For instance, by providing a topology that is coarser than each agent's one, the meet topology could be used as a representative collective topology.

It would also seem natural to require that the topologies on the state and event spaces should depend on each other. For instance, topologies on the event space could be given by specific extensions of the topological framework in line with the

[1] Note that similar considerations also arise in epistemic logical frameworks such as in DÉGREMONT and ROY (2012). Since the plausibility orderings in their framework could not only be defined on the states but also on the propositions, which are events from a semantic point of view, it could be of interest to analyze different – intuitive – ways of deriving plausibility orderings on propositions from the plausibility orderings on the states, or to more generally impose intuitive criteria on such orderings.

usual measure-theoretic structure the state space.[2]

Epistemically Plausible Topologies. Within the framework of topological Aumann structures, the topologies that represent epistemic features of a given underlying interactive situation or those that reveal particular agents's perception patterns of the event or state spaces are of specific interest.

For instance, in Section 4.4, we provided an epistemically plausible topology on the event space such that limit knowledge of rationality implies the solution concept strict dominance of order k. In Section 5.4, we described an epistemically-plausible topology on the event space such that limit knowledge is identical to Rubinstein (1989)'s notion of almost common knowledge (i.e., iterated mutual knowledge for some finite level m), and consequently, which induces a possibility for agents to limit-agree to disagree.

Epistemically plausible topologies on the state instead of the event space should also be considered. For example, consider the partition topology on the state space generated by the basis

$$\mathcal{B} = \left\{O \subseteq \Omega : O = \bigcap_{i \in I} \mathcal{I}_i(\omega) \text{ , for some } \omega \in \Omega\right\}.$$

Accordingly, every basic open set can be written as an intersection of the agents' possibility sets and interpreted as some kind of bundled, refined information of the agents. This topology represents the indistinguishability of worlds by all agents. Indeed, any two possible worlds ω and ω' are indistinguishable by all agents if and only if ω and ω' are not Hausdorff-separable, i.e., there do not exit two disjoint open sets O and O' such that $\omega \in O$ and $\omega' \in O'$. Equivalently, two possible worlds are distinguishable by some agents if and only if the two worlds are Hausdorff-separable. Hence, the partition topology conveys a notion of closeness between worlds that reflects precisely the informational indistinguishability between agents. Note that in this case, the closeness relation can be precisely determined. Indeed, the partition topology is in fact pseudometrizable with the pseudometric[3] $d : \Omega \times \Omega \to \mathbb{R}$ defined by $d(\omega, \omega') = k$, where k equals the number of agents being able to distinguish between ω and ω'. Consequently, the distance between any two worlds corresponds precisely to the number of minimal agents that are able to distinguish between these latter.

[2] Topologies on spaces of subsets of a given topological space X are typically defined in terms of the topology of X, such as the Hausdorff or Vietories topologies.

[3] A pseudometric space (X, d) is a set X together with a function $d : X \times X \to \mathbb{R}_+$ such that, for every $x, y, z \in X$:

- $d(x, y) \geq 0$;
- $d(x, y) = d(y, x)$;
- $d(x, z) \leq d(x, y) + d(y, z)$.

The pseudometric topology on X induced by d is the topology $\mathcal{T}_d$ induced by the open balls of d, i.e., the topology induced by the base

$$\{B(x, r) = \{y \in X : d(x, y) < r\} : \text{for all } x \in X \text{ and } r \in \mathbb{R}_+\}.$$

A topology $\mathcal{T}$ is pseudometrizable if there exists a pseudometric d whose induced pseudometric topology $\mathcal{T}_d$ coincides with $\mathcal{T}$.

A further example of an epistemically plausible topology is based on the notion of *evident knowledge event* or *common truism* (cf. Section 2.4). An event $T \subseteq \Omega$ is a *common truism* if and only if $CK(T) = T$. Intuitively, a common truism event is directly commonly known, i.e., it cannot occur without being commonly known, and can hence be understood as a reliable piece of information that all agents receive in public announcement or joined observation type situations. In fact, BINMORE and BRANDENBURGER (1990) already remarked that the set of all common truisms form a topology. This topology on the state space exhibits the property that two possible worlds ω and ω' are separable if and only if there exist two disjoint common truisms T and T' such that $\omega \in T$ and $\omega' \in T'$. The worlds are thus separated by mutually exclusive pieces of self-evident information which considerably distinguish them. Consequently, this topology expresses a notion of closeness between worlds based on the concept of indistinguishability via common truisms.

COUNTERFACTUALS. A notion of closeness between worlds is typically needed for theories of counterfactual reasoning. Hence, the enriched framework of topological Aumann structures could be used to model counterfactual knowledge and reasoning in set-based interactive epistemology. For instance, if some event E does not hold at the actual world ω, the reasoning of an agent i may depend on whether E is nevertheless close to what he actually considers possible, i.e., on whether the worlds contained in E are closer to the worlds $\omega' \in \mathcal{I}_i(\omega)$ than those contained in $\Omega \setminus (E \cup \mathcal{I}_i(\omega))$.

THE EPISTEMIC-TOPOLOGICAL APPROACH. Our topological approach to set-based interactive epistemology fits within the context of a series of papers involving topologies on the semantic structures, in order to introduce some notions of proximity which is not captured by the purely epistemic framework. For instance, MONDERER and SAMET (1996) proposed a topology on information structures defined in terms of the common belief that players have about the proximity of each player's information, and satisfying some important continuity properties of equilibria in games with incomplete information. KAJII and MORRIS (1998) described the weakest topology on probability distributions that is sufficient to restore the lower hemicontinuity between the probability distributions over the types and the corresponding equilibrium payoffs. DEKEL, FUDENBERG, and MORRIS (2006) defined the notion of a "strategic topology" on the Harsanyi-Mertens-Zamir universal type space which satisfies the property that two types are close if their strategic behavior is similar in all strategic situations.

Our topological approach to set-based interactive epistemology can be used to model some additional agents' perceptions of closeness between elements of the semantic structures, like worlds or events. This additional dimension enables to capture broader reasoning patterns of interacting or game-theoretic agents – involving some notion of "closeness" – which not only depend on the epistemic but also on the topological features of the interactive situation. These reasoning patters are therefore related to some subjective appreciation of the agents that some possible worlds, events or other features might seem to them closer or farther apart, without any logical justification. In a broad sense, this corresponds precisely to the modelling of some form of intuition.

Finally, in the long term, we hope that this enriched epistemic-topological framework could be used to better understand and model behaviors of actual agents, and hence, reduce the discrepancies between theoretical predictions and real-world situations (CAMERER (2003)).

BIBLIOGRAPHY

Robert AUMANN. "Agreeing to Disagree". In: *The Annals of Statistics* 4.6 (1976), pp. 1236–1239.

Robert AUMANN. "Backward induction and common knowledge of rationality". In: *Games and Economic Behavior* 8.1 (1995), pp. 6–19.

Robert AUMANN. "Common Priors: A Reply to Gul". In: *Econometrica* 66.4 (1998), pp. 929–938.

Robert AUMANN. "Correlated Equilibrium as an Expression of Bayesian Rationality". In: *Econometrica* 55.1 (1987), pp. 1–18.

Robert AUMANN. "Interactive epistemology I: Knowledge". In: *International Journal of Game Theory* 28.3 (1999), pp. 263–300.

Robert AUMANN. "Interactive epistemology II: Probability". In: *International Journal of Game Theory* 28.3 (1999), pp. 301–314.

Robert AUMANN. "Musings on Information and Knowledge". In: *Econ Journal Watch* 2.1 (2005), pp. 88–96.

Robert AUMANN. "On the Centipede Game". In: *Games and Economic Behavior* 23.1 (1998), pp. 97–105.

Robert AUMANN. "Reply to Binmore". In: *Games and Economic Behavior* 17.1 (1996), pp. 138–146.

Robert AUMANN and Adam BRANDENBURGER. "Epistemic Conditions for Nash Equilibrium". In: *Econometrica* 63.5 (1995), pp. 1161–1180.

Robert AUMANN and Sergiu HART, eds. *Handbook of Game Theory with Economic Applications*. 1st ed. Vol. 1. Elsevier, 1992.

Robert AUMANN and Sergiu HART, eds. *Handbook of Game Theory with Economic Applications*. 1st ed. Vol. 2. Elsevier, 1994.

Robert AUMANN and Sergiu HART, eds. *Handbook of Game Theory with Economic Applications*. 1st ed. Vol. 3. Elsevier, 2002.

Christian W. BACH and Jérémie CABESSA. "Agreeing to Disagree with Limit Knowledge". In: *Logic, Rationality, and Interaction - Third International Workshop, LORI 2011, Guangzhou, China, October 10-13, 2011. Proceedings*. Ed. by Hans van Ditmarsch, Jérôme Lang, and Shier Ju. Vol. 6953. Lecture Notes in Computer Science. Springer, 2011, pp. 51–60.

Christian W. BACH and Jérémie CABESSA. "Common knowledge and limit knowledge". In: *Theory and Decision* 73.3 (2012), pp. 423–440.

Christian W. BACH and Jérémie CABESSA. "Limit knowledge of rationality". In: *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2009), Stanford, CA, USA, July 6-8, 2009*. Ed. by Aviad Heifetz. 2009, pp. 34–40.

Christian W. BACH and Andrés PEREA. "Agreeing to disagree with lexicographic prior beliefs". In: *Mathematical Social Sciences* 66.2 (2013), pp. 129–133.

Christian BACH and Jérémie CABESSA. "Limit-agreeing to disagree". In: *Journal of Logic and Computation* (2016), Accepted.

Michael BACHARACH. "Some extensions of a claim of Aumann in an axiomatic model of knowledge". In: *Journal of Economic Theory* 37.1 (1985), pp. 167–190.

Jon BARWISE. "Three Views of Common Knowledge". In: *Proceedings of the 2Nd Conference on Theoretical Aspects of Reasoning About Knowledge*. TARK 88. Pacific Grove, California: Morgan Kaufmann Publishers Inc., 1988, pp. 365–379.

B. Douglas BERNHEIM. "Rationalizable Strategic Behavior". In: *Econometrica* 52.4 (1984), pp. 1007–28.

Kenneth G. BINMORE and Adam BRANDENBURGER. "Common knowledge and game theory". In: *Essays on the foundations of game theory*. Cambridge, Mass., USA : B. Blackwell, 1990, pp. 105–150.

Giacomo BONANNO and Klaus NEHRING. *Agreeing to Disagree: A Survey*. Department of Economics. California Davis - Department of Economics, 1997.

Tilman BÖRGERS. "Pure Strategy Dominance". In: *Econometrica* 61.2 (1993), pp. 423–430.

Tilman BÖRGERS. "Weak Dominance and Approximate Common Knowledge". In: *Journal of Economic Theory* 64.1 (1994), pp. 265–276.

Colin F. CAMERER. *Behavioral game theory. experiments in strategic interaction*. The roundtable series in behavioral economics. Princeton, N.J.: Princeton Univ. Press [u.a.], 2003.

Vincent P. CRAWFORD, Miguel A. COSTA-GOMES, and Nagore IRIBERRI. "Structural Models of Nonequilibrium Strategic Thinking: Theory, Evidence, and Applications". In: *Journal of Economic Literature* 51.1 (2013), pp. 5–62.

Cédric DÉGREMONT and Olivier ROY. "Agreement Theorems in Dynamic-Epistemic Logic". In: *Journal of Philosophical Logic* 41.4 (2012), pp. 735–764.

Eddie DEKEL, Drew FUDENBERG, and Stephen MORRIS. "Topologies on types". In: *Theoretical Economics* 1.3 (2006), pp. 275–309.

Eddie DEKEL and Marciano SINISCALCHI. "Epistemic Game Theory". In: *Handbook of Game Theory with Economic Applications*. Vol. 4. Elsevier, 2015. Chap. 12, pp. 619–702.

Lorenz DEMEY. "Agreeing to disagree in probabilistic dynamic epistemic logic". In: *Synthese* 191.3 (2014), pp. 409–438.

Martin DUFWENBERG and Mark STEGEMAN. "Existence and Uniqueness of Maximal Reductions Under Iterated Strict Dominance". In: *Econometrica* 70.5 (2002), pp. 2007–2023.

John D. GEANAKOPLOS and Heraklis M. POLEMARCHAKIS. "We can't disagree forever". In: *Journal of Economic Theory* 28.1 (1982), pp. 192–200.

Atsushi KAJII and Stephen MORRIS. "Commonp-Belief: The General Case". In: *Games and Economic Behavior* 18.1 (1997), pp. 73–82.

Atsushi KAJII and Stephen MORRIS. "Payoff Continuity in Incomplete Information Games". In: *Journal of Economic Theory* 82.1 (1998), pp. 267–276.

Willemien KETS. "Bounded Reasoning and Higher-Order Uncertainty". Working paper. 2014.

Willemien KETS. "Finite Depth of Reasoning and Equilibrium Play in Games with Incomplete Information". Working paper. 2014.

David K. LEWIS. *Convention: A Philosophical Study*. Harvard University Press, 1969.

David K. LEWIS. *Counterfactuals*. Blackwell Publishers, 1973.

Barton L. LIPMAN. "A Note on the Implications of Common Knowledge of Rationality". In: *Games and Economic Behavior* 6.1 (1994), pp. 114–129.

Lucie MÉNAGER. "Agreeing to Disagree: A Review". Working paper. 2012.

Paul MILGROM and Nancy STOKEY. "Information, trade and common knowledge". In: *Journal of Economic Theory* 26.1 (1982), pp. 17–27.

Dov MONDERER and Dov SAMET. "Approximating common knowledge with common beliefs". In: *Games and Economic Behavior* 1.2 (1989), pp. 170–190.

Dov MONDERER and Dov SAMET. "Proximity of Information in Games with Incomplete Information". In: *Mathematics of Operations Research* 21.3 (1996), pp. 707–725.

Stephen MORRIS. "Approximate common knowledge revisited". In: *International Journal of Game Theory* 28.3 (1999), pp. 385–408.

Stephen MORRIS. "Coordination, Communication, and Common Knowledge: A Retrospective on the Electronic-mail Game". In: *Oxford Review of Economic Policy* 18.4 (2002), pp. 433–445.

Zvika NEEMAN. "Approximating Agreeing to Disagree Results with Commonp-Beliefs". In: *Games and Economic Behavior* 12.1 (1996), pp. 162–164.

Zvika NEEMAN. "Common Beliefs and the Existence of Speculative Trade". In: *Games and Economic Behavior* 16.1 (1996), pp. 77–96.

David G. PEARCE. "Rationalizable Strategic Behavior and the Problem of Perfection". In: *Econometrica* 52.4 (1984), pp. 1029–1050.

Ariel RUBINSTEIN. “The Electronic Mail Game: Strategic Behavior under "Almost Common Knowledge."”. In: *American Economic Review* 79.3 (1989), pp. 385–391.

Dov SAMET. “Ignoring ignorance and agreeing to disagree”. In: *Journal of Economic Theory* 52.1 (1990), pp. 190–207.

Johan van Benthem and Darko SARENAC. “The Geometry of Knowledge”. Working paper. 2008.

Doron SONSINO. “"Impossibility of speculation" theorems with noisy information”. In: *Games and Economic Behavior* 8.2 (1995), pp. 406–423.

Robert C. STALNAKER. “A Theory of Conditionals”. In: *Americal Philosophical Quarterly* (1968), pp. 98–112.

Tommy Chin-Chiu TAN and Sergio WERLANG. “The Bayesian foundations of solution concepts of games”. In: *Journal of Economic Theory* 45.2 (1988), pp. 370–391.

H. Peyton YOUNG and Shmuel ZAMIR, eds. *Handbook of Game Theory with Economic Applications*. 1st ed. Vol. 4. Elsevier, 2015.