TOPICAL REVIEW • OPEN ACCESS

Towards a generalized theory comprising digital, neuromorphic and unconventional computing

To cite this article: Herbert Jaeger 2021 *Neuromorph. Comput. Eng.* **1** 012002

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
30 November 2020

REVISED
14 March 2021

ACCEPTED FOR PUBLICATION
24 March 2021

PUBLISHED
15 July 2021

TOPICAL REVIEW

Towards a generalized theory comprising digital, neuromorphic and unconventional computing

Herbert Jaeger*

Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence and Cognitive Systems and Materials Center (CogniGron), University of Groningen, The Netherlands
* Author to whom any correspondence should be addressed.

E-mail: h.jaeger@rug.nl

Keywords: neuromorphic computing, theory of computing, unconventional computing, non-digital hardware

Abstract

The accelerating race of digital computing technologies seems to be steering towards impasses—technological, economical and environmental—a condition that has spurred research efforts in alternative, 'neuromorphic' (brain-like) computing technologies. Furthermore, for decades, the idea of exploiting nonlinear physical phenomena 'directly' for non-digital computing has been explored under names like 'unconventional computing', 'natural computing', 'physical computing', or 'in-materio computing'. In this article I investigate coordinates and conditions for a generalized concept of 'computing' which comprises digital, neuromorphic, unconventional and possible future 'computing' paradigms. The main contribution of this paper is an in-depth inspection of existing formal conceptualizations of 'computing' in discrete-symbolic, probabilistic and dynamical-systems oriented views. It turns out that different choices of background mathematics lead to decisively different understandings of what 'computing' is. However, across this diversity a unifying coordinate system for theorizing about 'computing' can be distilled.

1. Introduction: why this is a good time to rethink 'computing'

Our modern societies are fundamentally shaped by digital computing (DC) technologies. There are a number of reasons why DC have grown into this majestic role:

Universality. Every information processing task that can be specified in a formal (first-order logic) description can be solved by a digital computer program. This is so because digital computers can emulate Turing machines and Turing machines can realize general theorem provers.

Transistors and wires. In mathematical abstraction, DC reduces to reading and writing 0's and 1's from and into hierarchical data structures. It is fully understood how this translates into transistor-and-wire based digital hardware architectures, and the corresponding microchip design and manufacturing technologies have reached astounding degrees of perfection.

Ease of use. A hierarchy of mutually cross-compilable programming languages—from hardware-specific assembler coding to graphical user interfaces for office software—allows users with all levels of expertise to exploit the potentials of digital computers.

Computing—cognition match. There is a pre-stabilized harmony between rational human reasoning and DC. A direct line of intellectual inquiry leads from Aristotle's syllogistic rules of reasoning through Leibniz, Boole, Frege and the early 20th century logicians to Turing, who in his groundbreaking paper [Turing 1936] still spoke of 'computers' as humans and of the physical states of a computing system as states of mind. Writing computer programs is just an exercise in logical thinking.

Unified theory. There is a unified, standardized body of DC theory, comprising automata theory, formal languages, the theory of computability and complexity, Boolean and first-order logic. This is documented in canonical textbooks and taught to computer science students in all universities in the same way, providing a conceptual and terminological common ground for a worldwide community of DC professionals.

In view of this intellectual transparency and practical empowerment, it is understandable that today 'computing' is largely identified with DC (=symbolic computing, =Turing computability, =running 'algorithms').

But progress rates of DC technologies are slowing down and seem to be approaching serious impasses:

Energy footprint. A widely recited estimate [Andrae and Edler 2015] claims that about 10% of the world's energy budget is due to DC technologies, with exponentially rising rates.

Miniaturization. Thermal and quantum noise and exploding investment costs for microchip fabrication may (or might not? See Murmann and Hoefflinger (2020)) prevent further downscaling of transistors in commercial microchip production—the 'end of Moore's law' [Waldrop 2016].

Toxic waste. Hardware replacement cycles are ever speeding up. Electronic waste 'is now the fastest-growing waste stream in the world' [Zhao *et al* 2019].

Software complexity and security. Software products are ever growing in size and complexity, perpetuating the *software crisis* since it was first acknowledged in the mid-1960s [Ebert 2018]. Operating systems which accumulate layer upon layer of code become incurably vulnerable to adversarial attacks. Given that critical segments of our modern world become permeated by complex software systems, we may be in for ruptures of societal functionality.

Such boundary conditions have led to a surge of explorations in 'brain-like', *neuromorphic computing* (NC) technologies. This field defines itself through the mission of transferring computational mechanisms used by biological brains to dedicated, increasingly often non-digital, electronic hardware systems. This goal was explicitly spelled out at least as early as in Mead (1990) and today has become an established engineering discipline with journals, conferences, study programs and textbooks [Zheng and Mazumber 2020]. NC seems a promising route to escape from some of the DC impasses:

Energy efficiency. Brains need only a minute fraction of the energy consumed by digital supercomputers for 'cognitive' tasks [Boahen 2017].

Unclocked parallelism. The inherent, complete parallelism of the brain's *in-memory computing* [Ielmini and Wong 2018] stands in stark contrast to the serial processing in DC systems where only a fraction of all available transistors are active at any time [Peláez 1990].

Cognitive-style computing. Artificial neural networks (ANNs) operate in ways which, at least at first sight, seem akin to human cognitive processing. They can, for instance, generate striking visual art [Olah *et al* 2017] or win against human world champions in the most cognitively demanding games [Silver *et al* 2016, Berner *et al* 2019].

Robustness and adaptability. Biological brains cope well with individual neuron death and to certain degrees even with extensive lesions. Their cognitive processing adapts to changing contexts, from reliable recognition of moving objects in changing lighting conditions to lifelong learning. While machine learning has only begun to understand such physical and functional robustness [Saunders *et al* 2019, He *et al* 2019a, Zhang *et al* 2019], brains are living proof that the hardware and functional brittleness of DC systems can be overcome.

NC research is blossoming, and there are solid indications that this enterprise is sustainable beyond hype cycle dynamics:

The deep learning revolution [ACM 2018] has manifested the powers of artificial neural information processing.

Large-scale digital neuromorphic microchips, developed by leading microchip manufacturing companies, emulate neural spiking for low-energy implementations of neural networks [Merolla et al 2014, Davies et *al* 2018, Neckar *et al* 2019]. This appears to be visible proof of the economical potential of NC.

Neuro-inspired algorithms [Lukoševičius and Jaeger 2009, Frémaux and Gerstner 2016] have been deployed on (partially) non-digital hardware [Indiveri and Liu 2015, Yousefzadeh *et al* 2018, Tanaka et al 2019, Neckar *et al* 2019].

Neuromorphic hardware solutions have been successfully developed for specific neuro-cognitive tasks, like artificial neural retina and cochlear implants that partially restore vision and hearing [Chuang *et al* 2014, Lenarz 2017], spiking neural camera sensor chips with integrated neural processing yield ultra-fast computer vision [Gallego *et al* 2020], or an entirely passive neuro-optical internet communication link prototype which needs no external energy supply [Freiberger *et al* 2017].

The advent of memristors in NC microchips [Yang *et al* 2013] has spurred material scientists to explore a wide range of physical nanoscale phenomena for computational exploits [Coulombe *et al* 2017, Torrejon *et al* 2017, Prychynenko *et al* 2018, Chen *et al* 2020, Mirigliano *et al* 2020, Prucnal *et al* 2020].

However, the neurosciences do not yet provide readily implementable blueprints for engineering novel physical computing systems. How the brain 'computes' is understood only in fragments. Foundational questions remain debated. For example, it has not been settled how 'symbols' or 'concepts' (addressable, stable representational entities) emerge from neural dynamics [Gross 2002, Durstewitz *et al* 2000, Baddeley 2003, Lins and Schöner 2014, Jaeger 2017, Besold *et al* 2017, Wernecke *et al* 2018]; how several such entities are coupled into composite entities [Buzsáki and Chrobak 1995, Slotine and Lohmiller 2001, Legenstein *et al* 2016]; how information streams can be dynamically routed in a brain [Olshausen *et al* 1993, Hoerzer *et al* 2014, Sabour *et al* 2017]; or how and in what sense 'information' is encoded in neural dynamics [Gerstner *et al* 1997, Panzeri *et al* 2017].

Stepping back from the daunting complexity of concrete biological brains, one may ask a meta-question which only at first sight looks naive: how could Nature ever 'invent' such magnificient systems?—For eons, biological evolution has been discovering, differentiating, optimizing and cross-coupling myriads of different biochemical, electrophysiological and anatomical phenomena, integrating them into that supremely adaptive, robust and balanced physical system that we carry in our heads. The structural, dynamical and functional complexity of this system's organization spans many orders of magnitude of spatial and temporal scales. Yet, throughout this breathtaking complexity, there is one grand unifying boundary condition: whatever phenomenon is exploited in a brain, it arises from the biochemistry and electrophysics of wet, soft biological tissue. Biological brains *must use only* that which is physically possible in a biological substrate—and they positively *do* use that.

Current artificial NC microsystems are, and future ones likely will be, manufactured from more enduring materials than wet neurons. Furthermore, engineers are already actively seeking to exploit physical effects that cannot occur in biological tissue, for instance optical, magnetic/spintronic, or even micromechanical effects [Prucnal *et al* 2020, Everhardt *et al* 2019, Leonov and Mostovoy 2015, Coulombe *et al* 2017]. If (i) a grand lesson to learn from Nature's brains is to exploit just everything which the available physical substrate offers, and if (ii) future hardware substrates will differ substantially from biological tissue, then it makes all sense to considerably widen the NC agenda, exploring how whatever physical phenomena in whatever material can be harnessed for 'computing'.

Exploiting 'the physics of the materials directly' [Zauner 2005] is absolutely not a new idea [Zauner 2005, European Commission Author Collective 2009, Stepney and Hickinbotham 2018, Horsman *et al* 2017, Adamatzky 2017a, 2017b, Stepney *et al* 2018]. This theme has been investigated for decades from different angles under a diversity of namings—for instance *unconventional*, *natural*, *emergent*, *physical*, *in-materio* computing—sometimes evoking a lasting echo like DNA computing [van Noort *et al* 2002], sometimes rather restricted to an academic niche like computing with fungi [Adamatzky 2018]. The term *analog computing* deserves a special mention. Traditionally it denotes approaches to assemble analog electronic circuits like adders or integrators into complex machines for realizing a hierarchy of computable functions, guided by formal theories that are shaped in the molds of DC theories of computable functions [Channon 1941, Moore 1996]. More recently however 'analog computing' has also been used in the entirely general sense to denote any non-digital approach [Mills 2008, Bournez and Pouly 2018]. I will use the term 'unconventional computing' (UC) in this article to refer to all of the above.

Summarizing, we can order DC, NC and UC in a row where at one end, DC has achieved an almost perfect grounding in a system of practically applicable and unifying theories, while at the other end UC can offer only the first inklings of theory-building. DC exploits an extremely narrow choice of physical phenomena, namely bi-stability of electronic circuits and memory devices, while UC explores an unbounded range of materials and nonlinear effects (figure 1). Finally, practical applications of DC are innumerable, while I am not aware of a single UC system in everyday practical use. NC lies between DC and UC in all these respects.

To avoid misunderstandings I point out four things that I do *not* include when I speak of UC. First, I am not concerned with *pancomputationalism* where formal concepts from symbolic computing are invoked to describe and explain the physical world [Lloyd 2013]. Second, I am not dealing with *physics of computation*, a field that explores the physical boundary conditions of DC [Wolpert 2015]. While these researches give inspirations for mathematical formalizations of complex, self-organizing, physical pattern formation [Zuse 1982, Wolfram 2002, Fredkin 2013], one must be aware that these traditions are immersed in the DC understanding of 'computing' as discrete symbol manipulation processes. Third, I will not consider the question of *which physics is computable* in the sense of perfect simulations of physical phenomena on (future, non-standard) computing machines—a deep question brought up by Feynman (1982). Finally, I see the wider fields of UC as hardly intersecting with *quantum computing*, which has matured into a discipline of its own standing and only peripherally connects with other UC research, as for instance in quantum reservoir computing [Nakajima *et al* 2019]. While quantum computing certainly exploits 'the physics of the materials directly', it rests on a single physical phenomenon (indeterminacy of quantum states) and focusses on a specific subset of classical

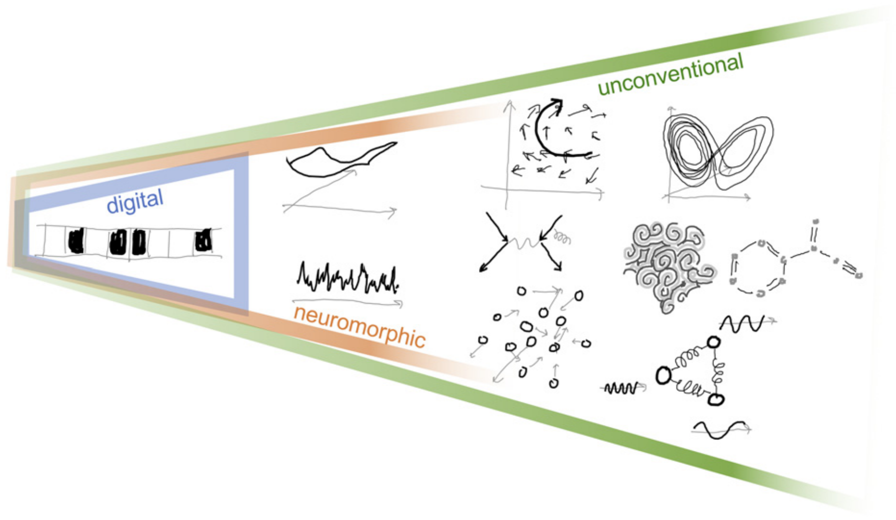

Figure 1. The scopes of digital, neuromorphic and UC in a napkin drawing. DC exploits only binary switching. NC furthermore standardly uses real-valued functions and stochasticity, but occasionally more. UC is open to a wide spectrum of physical effects which today are modeled with an equally wide range of formalisms.

DC tasks. Quantum computing thus operates in a narrowly defined intersection of the physical and the computational domains. Of course this niche is of potential great practical importance; but this article aims at a generalization of DC for exploiting a not *a priori* bounded wealth of physical effects—a twofold opposite of quantum computing.

2. Staking out the landscape of 'computing' theories

It seems uncontroversial that DC, NC, and UC alike concern 'computing' systems. If scientists feel that it is right to use the same word for a multiplicity of physical phenomena, then there should be a formal theory that brings the underlying unity to the surface.

The ground seems already amply covered by introductions, surveys, methodological frameworks and philosophical inquiries into the nature of 'computing'. What would still be missing, what could I possibly add by writing yet another article? Let us take a closer look at what is already there.

Among the three domains of DC, NC and UC, the first one undoubtedly commands the best worked-out body of theory. The formal theory of DC has crystallized into canonical textbooks which are taught to computer science students worldwide in essentially always the same coverage and terminology. I find it important to point out that the powers of the DC paradigm do not emerge from a single formal theory or model. Besides the theory of Turing machines, which could be mistaken as *the* theory of DC, there are other formalisms, models and theories that are just as essential for the practical manifestations of DC. They include the theories of automata, formal grammars and languages, programming languages and compiler design, computability and complexity theory, Boolean and first-order logic, other logics and metalogical frameworks. These theories, models and formalisms are *tightly and transparently interrelated*. Their totality can be likened to an orchestra which instruments all professional DC activities from device engineering to microchip fabrication technologies, from circuit design to computer architectures and communication networks, from programming language development to human–computer interfacing, from databases to internet services, from beginners' programming exercises to software engineering and use-case specification frameworks and all the rest.

One of the challenges faced by NC is that such a theories (plural) orchestra needs more time to grow than this field so far has had. As a consequence, always renewed ad hoc efforts are still needed to transfer any single novel NC technique from lab A to lab B, let alone to a wider user community. High-investment efforts to define systematic, theory-guided workflows in the NC domain have just been launched [Zhang *et al* 2020] and demonstrate that there is a serious demand. I can share a vivid first-person experience in that regard. In a collaborative NC research project my group was charged with getting an analog spiking neuromorphic microprocessor board (the SYNAP-SE [Moradi *et al* 2018]) to function as an online human heartbeat monitoring system. My team members and I had backgrounds only in computer science, machine learning and mathematics. The available technical documentation and supporting literature, written by electronic circuit engineers and microchip designers, was hardly understandable for us. The interfacing between the analog SYNAP-SE and our digital host computers forced us to learn about digital–analog conversions, using oscilloscopes and finding out that the USB cable had to be shorter than 25 cm. We only mastered the technical handling of the microprocessor after one year of labor, and furthermore we had to develop novel mathematical models that allowed us to transfer algorithms, developed in digital simulation, to the analog SYNAP-SE [He *et al* 2019b]. These transfer models were specific both to the analog target system and to our classical source algorithms and would not be easily re-usable in other NC set-ups. All of this was an impressive demonstration that NC is still an eminently interdisciplinary endeavor, with all the implied complications and delays. Accordingly, the most recent NC textbook [Zheng and Mazumber 2020] imports materials from machine learning, neuroscience and microchip technology. A large number of formal models, empirical findings and prototypical examples from across these areas is presented which represent '*the latest developments and trends in energy-efficient NC*'. But in contrast to DC textbooks, these items are not yet connected to each other through transparently interrelated theory links.

The UC landscape is even more diverse than the one of NC. This is only natural because UC is curious about *any* kind of physical phenomenon that could be harnessed for computing, while NC efforts are oriented toward a single role model, the brain. A variety of surveys, conceptualizations and classification schemes for UC have been proposed [Harnad 1994, de Castro 2006, Mills 2008, Burgin and Dodig-Crnkovic 2013, Stepney 2017, James *et al* 2017, Bournez and Pouly 2018]. However, in my opinion, where general unifying theory frameworks have been proposed, as in MacLennan (2004), Mills (2008), Horsman *et al* (2017) or Stepney and Kendon (2020), they are too closely leaning on paradigmatic assumptions inherited from DC (in particular, identifying 'computing' with calculating function values) and/or formulated on an abstraction level that is too high to give concrete guidance for system design and use. Two recent collection volumes [Adamatzky 2017a, 2017b] illustrate the fertility of the field on the one hand, but on the other hand testify to the unconnectedness of its manifold contributions. The 31 chapters in the *theory* part (volume 1) treat 31 individual sectors of the NC landscape and none ventures to propose a unified theoretical framework. Stepney and Hickinbotham (2018) list a number of existing mathematical formalisms that are tailored to specific subsets of material phenomena or computational functionalities. Their final dictum still seems valid, namely that an '*over-reaching formalism… may be desirable*'. All in all, the UC community is far away from a universally agreed body of theories which would systematically interrelate all aspects and activities in UC research across materials, nonlinear phenomena, device engineering techniques, task specifications, use cases and user interfacing methods, or professional training programs.

This article rests on the assumption that it is possible to develop a body of interrelated formalisms, models and theories that can stand side by side with what we see in DC textbooks, and which can practically enable computational engineering workflows to exploit the unlimited richness of physical phenomena that we meet in UC collection volumes. This would give rise to a *general computing* (GC) science which would include DC and NC as special cases.

Previous proposals for a generalized theory of computing attempted to formulate exactly that, *a* generalized theory—singular. In contrast, I start from acknowledging that an entire spectrum of formalisms, models and theories will be needed to cover the many steps of engineering pipelines from materials and device engineering to user interface design—like looking through a prism, splitting white light into its rainbow components. As a first step toward a system of interrelated theories for GC I will inspect in some depth several established views on 'computing' from DC, NC and UC. The aim of this section is to identify component theories that are homologues of each other across these different domains. I hope that a clearer recognition of what sorts of sub-theories are needed in any complete account of 'computing' will be helpful to actually work them out.

2.1. The DNPU device: a digital–neuromorphic–unconventional chameleon

To add some flesh and intuition to the theoretical investigations which will follow, I first want to present a concrete physical device (or is it a circuit? or many?) which can be viewed from all three perspectives: DC, NC and UC.

This *Dopant Network Processing Unit* (DNPU) was designed, fabricated and made to compute in various ways by the nano-electronics group of Wilfred van der Wiel at the University of Twente. It consists of a boron- or arsenic-doped silicon surface patch around which a number of electrodes are placed in a circle (figure 2). In the exemplars discussed here there were eight electrodes, but their number is arbitrary. When voltages are applied to the electrodes, electrons or holes hop between neighboring dopant atoms. In conjunction with electrostatic interactions, the resulting currents between the electrodes acquire complex nonlinear dependencies on the electrode voltages. In typical experiments, one electrode is grounded and its current is used as the *output* response (configurations with several outputs are also possible). The remaining seven electrodes can be freely assigned roles as *input* or as *gating* terminals.

In the experiments that have been reported so far, the gating voltages were kept constant and the input voltages were switched between different constant value combinations. When the former are switched to a new input voltage pattern, after a short relaxation time of a few nanoseconds [Chen *et al* 2020] the output

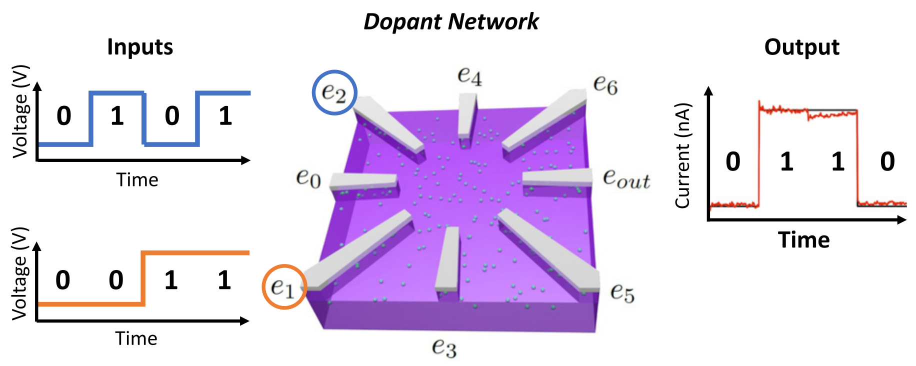

Figure 2. Schematic of the DNPU device, here realizing an XOR gate. The central area between the electrodes has a diameter of about 300 nm (Reproduced with permission from Ruiz Euler *et al* (2020)).

current stabilizes. The device thus realizes an input (voltage) pattern to output (current) value—a *function*. By suitable settings of the gating voltages a wide range of functions has been experimentally demonstrated:

- All binary Boolean functions [Euler *et al* 2020].
- Nonlinear nonmonotonic functions that are abstractly analog to neural excitation/inhibition characteristics [Chen *et al* 2021].
- A four-input function that recognizes all possible binary 4-bit patterns and a function that realizes a circular decision boundary in the unit square [Euler *et al* 2020]. These are classical demo tasks for small multilayer neural networks, revealing that a single DNPU unit can emulate such networks.
- Ten DNPU units, joined in parallel to provide the single processing layer of a 'neural network', realized an image classification 'neural' network, which on the standard MNIST benchmark achieved a 96% testing accuracy in a simulation study [Ruiz Euler *et al* 2020].

The first item in this list qualifies DNPUs as generic digital switching devices and the remaining ones recommend it as a space- and energy-efficient building block in neuromorphic microchips. The class of functions which can be realized by the DNPU awaits its complete characterization. A hint at the richness of this function class is that a fairly large deep neural network (5 layers with 90 neurons each) was needed to model this function space on the basis of training data sampled from experimental measurements [Euler *et al* 2020].

In earlier work from the same group, a similarly structured and sized eight-electrode device, where the active substrate was a gold nanoparticle cluster instead of a doped silicon surface, likewise could realize all binary Boolean functions [Bose *et al* 2015]. Working with different materials and physical mechanisms exemplifies the UC attitude of 'exploiting the physics of the materials directly', although in the research described above the considered computational paradigms were still the ones from DC and NC.

In future work van der Wiel plans to leave the safe harbor of DC and NC and set forth into unfathomed waters of UC. So far DNPU units have been explored as functions, allowing them to stabilize after resetting input voltage patterns. Energy efficiency and processing speed/bandwidth could be enormously increased if DNPUs were fed with input signals whose time constants would be so small that the DNPU will always operate transiently at its native nanosecond timescale. The gating signals, which so far have been constant, could also become time-varying, up to timescales as fast as the 'input' signals which would lift the distinction between input and configuration/control signals. Such *wild* [Jaeger 1998] dynamics can be put to use with methods of reservoir computing [Tanaka *et al* 2019] as a first step, but in the long run would need a GC theory to domesticate their full powers.

2.2. Approach

Figure 3 shows the main conceptual components that I want to explore in some detail, relating them to each other. I want to dissect the physical reality (bottom half of figure) into the computing hardware systems (α) and the physical environment (γ) they are embedded in and in which they should serve some purpose. The interface boundary (β) between these two comprises the physical signals that are exchanged between the computing systems and their environment. These three segments of physical reality are mirrored in the non-physical domain of mathematical abstraction by corresponding formal representations of the physical computing system, providing information processing models (a), input/output (I/O) data models (b), and formal models of outward physical realities (c). Figure 3 fills each of these six component boxes with a number of suggestive examples.

Given a specific computing system with its environment and/or the formal models thereof, it is a matter of convention where the interface boundary is placed. For instance, when discussing human brains, the visual

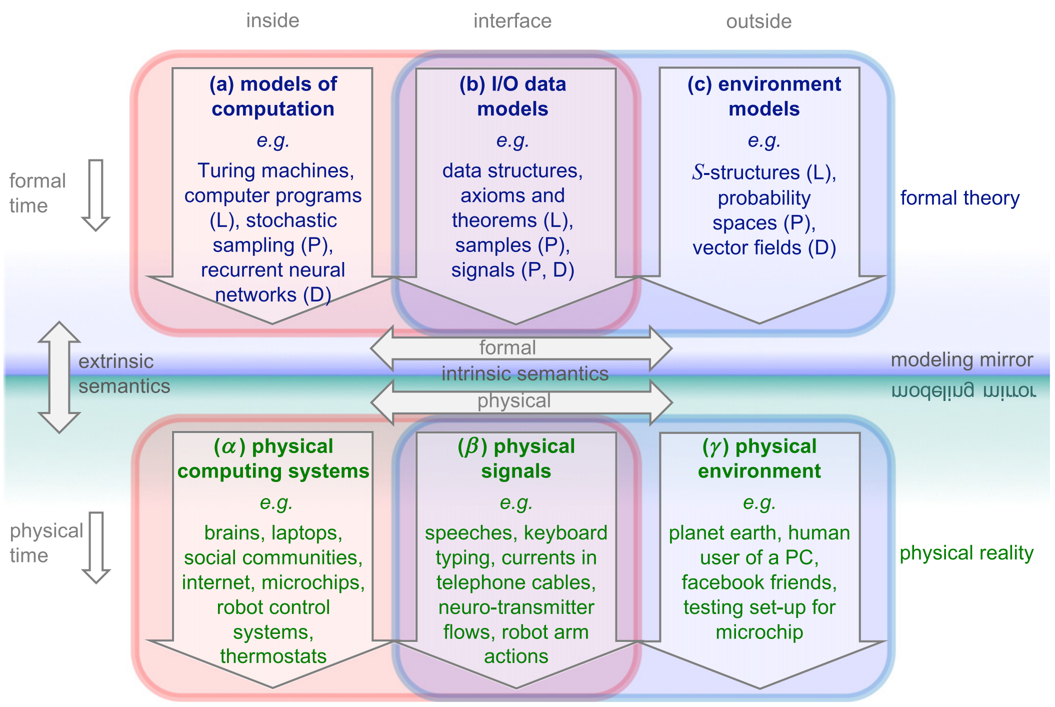
inside
interface
outside
(a) models of computation
e.g.
Turing machines, computer programs (L), stochastic sampling (P), recurrent neural networks (D)
(b) I/O data models
e.g.
data structures, axioms and theorems (L), samples (P), signals (P, D)
(c) environment models
e.g.
$\mathcal{S}$-structures (L), probability spaces (P), vector fields (D)
formal theory
formal time
extrinsic semantics
formal
intrinsic semantics
physical
modeling mirror
(α) physical computing systems
e.g.
brains, laptops, social communities, internet, microchips, robot control systems, thermostats
(β) physical signals
e.g.
speeches, keyboard typing, currents in telephone cables, neuro-transmitter flows, robot arm actions
(γ) physical environment
e.g.
planet earth, human user of a PC, facebook friends, testing set-up for microchip
physical reality
physical time

Figure 3. Schema of the six main elements (a)–(c) (formal) and α, β, γ (physical) in my navigation map to locate formal theories, which here are loosely characterized as logic-based (L), probabilistic (P) or dynamical systems oriented (D).

sensory input signal boundary could be placed at the interaction between photons and photoreactive receptor molecules inside retinal photocells, or it could be assigned to the spike trains sent through the optical nerve.

A core question that must be implicitly or explicitly answered by any GC theory is what makes a system ‘compute’. The classical answer, which is also employed in all accounts of UC that I am aware of, is to define ‘computing’ in the footsteps of DC. According to this conceptualization, computing involves several steps, namely first encoding computational ‘tasks’ into a formal input, which is then physically entered into a physical computing system (a step that passes the *modeling mirror* between abstract formalism and physical reality and involves un-formalizable pragmatics), then let the physical system physically evolve until some physically observable halting condition is met, then read off the physical system’s state into a formal output (again, this step crosses the modeling mirror and involves pragmatics), then decode the output into the task’s solution.

I do not think that this view of ‘computing’ is suitable for a generalized theory, one reason being that biological brains (which a generalized theory should cover as special cases) operate in a continuous temporal input-to-action interaction with the environment which is quite different from the time-decoupled single-input-to-single-output transformation functionality expressed in the classical view. I believe that brain-like systems will often be operating in a modality where they are *entrained* to a driving input stream. This is different from DC accounts of *real-time processing*, where the processing system is executing a series of subtasks fast enough to stay on track with the incoming processing demands, like in the model of interactive Turing machines [van Leeuwen and Wiedermann 2001]. During the computational operations which solve the individual subtasks the processing machine decouples itself from incoming input. A more general view of ‘computing’ is needed which natively includes entrained input stream processing.

While I cannot give a complete set of necessary and sufficient conditions, I think that the following five are necessary and quite informative for a system to be called ‘computing’ in an defendable way:

(a) ‘Computing’ systems must be *physically realizable.* It is not enough to only have some abstract mathematical model of ‘computing’.

(b) ‘Computing’ requires a *temporal evolution* of the system that computes—this is why the temporal arrows are so prominent in figure 3.

(c) ‘Computing’ involves that ‘information’ feeds into, and comes out of the computing system: computing systems must be *open* systems—this is why I devote a special conceptual place (β) to the interface boundary between the computing system and its environment.

(d) ‘Computing’ operations going on in a computing system should be *cognitively interpretable*, being relatable to some aspect of human cognition. This requirement is authorized by 2370 years of intellectual labor, starting with Aristotle’s syllogistic logic and not ending with Alan Turing who explicitly equated the symbolic configurations in Turing machines with 'states of mind' [Turing 1936]. Also, the NC motif to 'learn from the brain' connects 'computing' directly with cognition.

(e) 'Computing' is *meaningful*, that is, some *semantic* account of 'what' is computed should be possible—indicated in the figure by the two semantics arrows (to be detailed below).

A formal theory needs a mathematical language to be expressed. What mathematical language is used is often tied to the primary disciplinary background of the modeler. Researchers with a background in computer science or AI tend toward discrete-algebraic and logic-based formalisms; cognitive scientists may prefer probabilistic (Bayesian) frameworks; physicists tend toward dynamical systems based modeling languages. These are marked L, P, D (for logic, probability, dynamical systems) in figure 3. The adopted background mathematics determines which phenomena can be modeled and which not. The chosen background mathematics often reflects a deep-rooted personal or community perception of physical or cognitive reality. This may evoke hefty epistemological debates, like the decades-long *physical symbol systems hypothesis* controversy about the physical-neural reality of 'symbols' in cognizing brains [Newell and Simon 1976, Fodor and Pylyshin 1988, Brooks 1991, Pfeifer and Scheier 1999, Laird *et al* 2017, Buongiorno 2019]. Entire modeling schools (like cybernetics, classical AI, or Bayesian agent modeling) have grown around a fundamental commitment to a specific background mathematics, and have helped to advance the very mathematical methods in return. In the final section I will argue that neither L, P or D type mathematics alone is appropriate for GC.

The two directions bottom-top and left-right in figure 3 can be regarded as *semantic* axes. The bottom–top axis, which I will call *extrinsic* semantics, connects formal models with their physical counterparts, crossing the modeling mirror between the ontological domains of the formal and the physical. Since the physical end of this axis is just the physical reality, extrinsic semantic pairings between models and reality cannot be formalized. Instead, this link is established by social conventions and practical routines. Students learn to connect the two ontological domains by practical exercises and epistemologists work out what it means that a formal model is 'valid' or 'adequate' in some sense or other. The left-right axis is the bridge between what happens or is 'represented' inside a computing system and what happens or 'exists' in its outside environment. This bridge is instantiated twice, within the formal domain where it connects segments **a** and **c**, and within the physical domain where it connects α with γ. Because these semantic relations are spanned within their respective ontological domains I will call them *intrinsic* semantics. Formal intrinsic semantics can be mathematically formalized because on both ends **a** and **c** there are mathematically defined constructs. This will be an important theme in the detailed discussion given below. The physical intrinsic semantics comprises all the real-world interactions between a physical computing system α and its physical environment γ, mediated by physical signals. For example, what happens inside a thermostat (in segment α) interacts in physically lawful ways with what happens in the room (in segment γ) where it has been installed.

The study of semantic relations between symbols and their 'meaning' is an ancient and deeply thought-out theme in philosophy. I am only superficially acquainted with this body of thought and my separation of semantic relationships into extrinsic and intrinsic ones is certainly a rather naive simplification, and my division of everything into merely two ontological domains (formal and physical) is surely equally naive. I found the simple ontological bipartition of everything and the resulting intrinsic/extrinsic separation nonetheless quite helpful in organizing my thinking, and I make use of it when I explain logic to computer science students [Jaeger 2019b].

The schema shown in figure 3 suggests a simplicity than does not exist. Historical theory-building processes are in many respects like biological evolution, leading to different solutions in different niches (scientific disciplines and communities). Methodological differences can be drastic, as between the humanities and the natural sciences. But even within and between the neighboring disciplinary strands of computer science, AI, and machine learning we find distinctively different ways of formalizing and theorizing about 'computing'. Before one embarks on GC theory development it is instructive to inspect existing 'computing' theories and formalisms at a finer resolution than in figure 3.

The words 'theory', 'formalism' and 'model', which I will use a lot, should be handled with care. I understand them in the following way. A *formalism* is a set of more or less rigorously specified conventions of what kind of formal expressions one may write down. Examples are the formalism of first-order logic, the formalism of probability theory, or the formalism of ordinary differential equations (ODEs). Also every programming language is a formalism. Formalisms are languages. Logicians and computer scientists often specify formalisms in complete accuracy through grammars, while non-logician mathematicians and physicists rely on informal conventions (which nonetheless are binding and well-understood). A *model* is a specification of a particular piece of reality, written down in the notation of a specific formalism. Finally, in the strict sense of mathematical logic, the word *theory* denotes the set of all theorems that can be proven from a set of axioms, like the theory of groups. Outside logic, the word is used in a wider and less strict sense, often comprising an entire collection

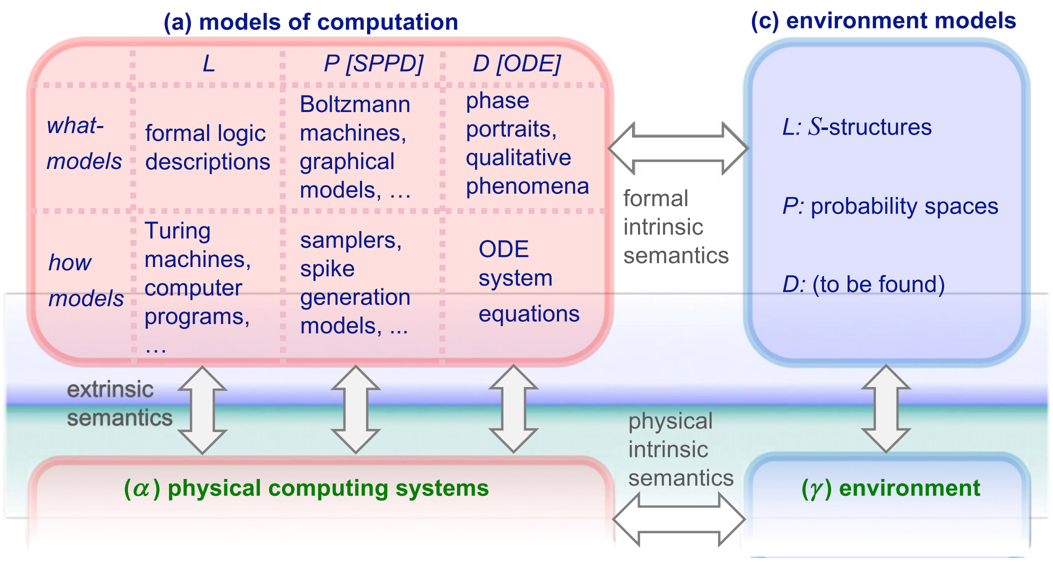
(a) models of computation
L
P [SPPD]
D [ODE]
what-models
formal logic descriptions
Boltzmann machines, graphical models, …
phase portraits, qualitative phenomena
how models
Turing machines, computer programs, …
samplers, spike generation models, …
ODE system equations
formal intrinsic semantics
(c) environment models
L: S-structures
P: probability spaces
D: (to be found)
extrinsic semantics
physical intrinsic semantics
(α) physical computing systems
(γ) environment

Figure 4. Zooming into figure 3, giving an overview of themes treated in this article in the perspectives of symbolic/logic (L), sample-based probabilistic [P (SPPD)], and ODE-based dynamical systems [D (ODE)] modeling.

of intended models, specifications of observation procedures, and predictions and hypotheses, like in the theory of quantum mechanics or the theory of supply and demand. I generally adopt this second usage and say 'deductive theory' when I mean the first one.

A complete coverage of all computing-related formalisms and models in the symbolic-logic (L), probabilistic (P), and dynamical systems oriented (D) modeling domains is infeasible and I have to confine the discussion to selected fragments of the total modeling cosmos. The theory landscape in symbolic/DC is canonically worked out and its essentials fit into one textbook which all students of computer science have to digest. I will be able to give an almost comprehensive account here. This is not the case for the P and D domains whose theory-building landscapes are very diverse and unifying meta-views are not available. In the probabilistic modeling domain I will consider only formalisms and models where probability distributions are represented and 'computed' through sampling mechanisms, and use the acronym SPPD ('sampling-based processing of probability distributions') to characterize this kind of formalisms and models. With regards to dynamical systems modeling I will only inspect the fragment where ODEs are used and speak of ODE formalisms and models.

Within each of the L, P and D domains I will find it helpful to distinguish between two kinds of formalisms and models.

The first kind captures what one could call the 'mechanics' of computing processes. I will call these *how*-formalisms and models. In the three domains this will be the abstract models of algorithms and the programming languages (L), sampling algorithms and physics-oriented models of certain stochastic processes that are interpreted as sampling processes (P), and ODE models of brains and analog computing machines (D).

The second kind comprises formalisms and models whose core constructs capture cognitively interpretable aspects of human information processing. I will refer to these as *what*-formalisms and models. In the L domain this will be logical inferences on the basis of symbolic expressions, and the what-formalisms are the formalisms of symbolic logic, of which there are many. In the P domain the core cognitively interpretable constructs are probability distributions. In probabilistic accounts of cognitive processing (in theoretical neuroscience, cognitive science, AI and machine learning), probability distributions are widely considered as a useful and appropriate mathematical correlate of *concepts*. Probabilistic what-models there are as many as there are ways to conceive probabilistic conceptual reasoning, but all of these models are expressed in a single formalism, namely the canonical textbook formalism of mathematical probability theory. Finally, in the D domain, the cognitively interpretable mathematical constructs that can be defined on the basis of ODEs are qualitative geometrical constructs like *fixed points*, *attractors*, *bifurcations* and many more. In the cognitive and neurosciences, such geometrical items have become perceived as correspondents of a diverse range of cognitive phenomena. Similar to what we find in probabilistic modeling, what-models there are many, but they are all written down using the same basic ODE formalism. Figure 4 shows a zoom into segment **a** of figure 3.

I will discuss L, P, D formalisms with regards to the themes

(a) Interrelationships between formalisms,

(b) Formal intrinsic semantics,

(c) Formal time, and
(d) Hierarchical organization of formal constructs within a formalism.

I leave a discussion of other, likewise relevant themes for other occasions. In particular I will not consider aspects of

- Functionality—what tasks or problems are solved by ‘computing’,
- Pragmatics—whether one can ‘program’ a computing system or one has to ‘train’ it; how can ‘users’ interact with the ‘computer’, and what are the ‘users’ in the first place,
- Information—how ‘encoding’, ‘uncertainty’, ‘precision’ are formalized,
- Learning, adaption, calibration, stabilization,
- Growth—extensibility of models and physical systems,
- Complexity—what are measures and limits of a system’s ‘computing power’,
- Intra-model communication—what are ‘signals’ and how are they propagated inside a computing systems,
- Formal space—how are what- or how-models metrically or topologically structured, and how this relates to formal time.

The four themes which I do discuss are key coordinates for theorizing in digital computer science. But upon inspection of non-digital information processing models in the P and D domains it will turn out that these themes are also instructive entry points to study non-DC concepts, and that they can be worked out in very different ways from what we are accustomed to in DC.

2.3. Findings

This subsection constitutes the main substance of this paper. It is very long, yet too short for the level of detail that would make it self-contained. In many places I must rely on the reader’s goodwill to follow. At the end (section 2.4) I try to tie everything together in a high-level summary that can be read stand-alone, so the reader may choose to jump forward directly.

2.3.1. DC formalisms and models

I begin my detailed investigation with a discussion of DC formalisms and models. A first overview: the how-formalisms include abstract descriptors of algorithmic processing which are mainly used for theoretical analysis (like the Turing machine or certain grammar formalisms), as well as formalisms which are destined for practical use, most importantly programming languages. Some abstract how-formalisms are useful also for practical programming, in particular lambda calculus (which directly spins off the so-called functional programming languages) and the random-access machine model (which is close to assembler programming languages).—The what-formalisms are logic formalisms, with Boolean logic and first-order logic the standard textbook representatives. What-models are sets of logical formulas that specify what a program should compute. Such specifications are needed when one wants to formally verify that a written computer program actually fulfills its purpose—an important objective in many application domains. A special case are the so-called declarative programming languages, like Prolog or functional programming languages, which can be dually regarded as how- and what-formalisms and allow the user to write programs directly in terms of ‘what’ tasks shall be computed by the program. All of the above is explained in any textbook of theoretical computer since, such as my own online lecture notes [Jaeger 2019a, 2019b].

There are further formalisms that belong to the wider circles of DC theory which do not fall into the how- and what-classes that I mentioned here. For example, I exclude from consideration formal communication protocols between different computers. The formalisms that I mentioned however can be considered the bone and marrow of theoretical computer science.

Take-home message 1: the separation between how- and what-formalisms and models is not entirely clear-cut. Some how-formalisms can climb to levels of cognitively interpretable abstraction that they could also be considered what-formalisms.

Formalism interrelations. Both how- and what-models in DC are noted down using tools from discrete mathematics, being based on finite alphabets of symbols. How-models describe computing processes in terms of step-wise, rule-based updates of compound symbol configurations. There is a well understood hierarchy of how-formalisms, defined by different levels of *expressiveness*. A formalism A is at least as expressive as a formalism B if every input–output task that can be solved with models formalized in B can also be solved with a model formalized in A. This can be equivalently stated in a ‘syntactic’ way, as follows. Consider a model $\mathcal{B}$ formalized in B (think of a computer program $\mathcal{B}$ written in a programming language B). Then there exists a model $\mathcal{A}$ formalized in A and an encoding function $\tau_{\mathcal{B}<\mathcal{A}}$ which translates any symbolic configuration b that can occur in a ‘run’ of $\mathcal{B}$ into an $\mathcal{A}$-configuration a (that is $\tau_{\mathcal{B}<\mathcal{A}}(b) = a$) such that, when $b \rightarrow b'$ is a configuration update step that can occur when running $\mathcal{B}$, there is a sequence of update steps in $\mathcal{A}$ of the form $\tau_{\mathcal{B}<\mathcal{A}}(b) \to a_1 \to \cdots \to a_n \to \tau_{\mathcal{B}<\mathcal{A}}(b')$. In technical terminology: every run of $\mathcal{B}$ can be *simulated* by $\mathcal{A}$. This leads to a hierarchy of classes of formalisms, where within each class, all formalisms have the same expressivity (they can mutually simulate each other), and where formalisms in less expressive classes can be simulated by formalisms in more expressive classes but not vice versa. This hierarchy, often referred to as the *Chomsky hierarchy*, is explained in every computer science textbook and is a pillar of theoretical computer science. The most expressive class known today is the class of Turing-equivalent formalisms. It comprises all practically used programming languages and many abstract mathematical formalisms, including of course the formalism of Turing machines. According to the *Church–Turing hypothesis*, no more expressive kind of how-formalism exists.

While all Turing-equivalent how-formalisms are equally expressive, it is nonetheless useful to arrange them, within their class, in an abstraction hierarchy, with 'low-level' formalisms close to the fine-grained 0–1 switching of transistors in digital hardware (for instance assembler programming languages), and with 'high-level' formalisms (like declarative programming languages or Microsoft Excel) closer to the human programmer's cognitive representations of what it means to process information. When one writes a program in a high-level language and starts it from a high-level user interface, behind the scene it becomes level-wise *compiled* into lower-level ones until a version specified by the microchip manufacturer is reached, which can directly address the hardware α, crossing the modeling mirror shown in figure 3.

Trained computer scientists can easily create new, again equivalent how-formalisms tailored to their particular needs. This flexibility and transparency to move up and down in the abstraction hierarchy is unique in DC and is one reason for the current intellectual and practical dominance of DC over all other approaches to 'computing'.

Take-home message 2: DC how-formalisms can be systematically ordered by their expressiveness. A formalism *A* is more expressive than a formalism *B* when all *B* models can be simulated by *A* models. Within the class of most expressive formalisms—the Turing-equivalent ones—one can furthermore roughly order them according to how close they are to 'low' levels bordering to the physical 0–1-switching of digital circuits, or to 'high' levels that are more interpretable in terms of human cognitive operations.

As to what-formalisms (formal logics), we find a wealth of different logics (plural) in the DC domain. Besides the two logics which are standardly taught to students of computer science (Boolean and first-order logic), there is also second-order logic, trimmed-down fragments of first-order logic, and indeed infinitely many more. A core theme of contemporary theoretical computer science is to work out *logical frameworks* [Rabe 2008] to navigate in this richly populated landscape. A key ordering principle is *logic translations*, that is meta-formalisms which allow one to specify how one logic A can 'express' everything that another logic B can. This is only superficially similar to the expressiveness hierarchy we saw for Turing-equivalent how-formalisms. The expressiveness hierarchy for how-formalisms is defined on the basis of the *syntax* of symbolic configurations, while for logic translations a *semantical* account is needed of what the symbolic expressions that can be written down in a logic formalism mean. This renders the study of expressiveness relations between what-formalisms much more involved than it is for how-formalisms. Research in this area is far from being canonically completed.

Take-home message 3: what-formalisms in DC are formal logics, of which there are many. Like how-formalisms they can be ordered according to their expressiveness, but here 'expressiveness' is defined in terms of formal intrinsic semantics, not in terms of syntactically describable transformations of symbol configurations.

Formal time surfaces in distinctively different ways in how- versus what-formalisms. For how-formalisms the story is quickly told. The formal, discrete update rules acting on symbolic configurations become ultimately mirrored in the physical clock cycles of digital microchips. For what-formalisms it is more difficult to understand time. Explaining how time arises in what-formalisms is intimately connected to **intrinsic semantics**. What-formalisms cast and connect the segments **a**, **b**, **c** from figure 3 with the mathematical constructs of formal logic, often first-order logic. In logic-based what-formalisms, the model **c** of the world γ around the physical computing system α is usually characterized in terms of certain set-theoretic structures, called *S-structures*, where *S* stands for the symbol alphabet of the formalism. Working out this universal connection between logic and set theory early in the 20th century was a milestone for mathematics. Logical inference (that which happens, for instance, in Aristotle's syllogisms) became formalized through inclusion relations between classes of *S*-structures. This modern mathematical re-construction of logical inference reached its final form in the work of Tarski (1936). Neither *S*-structures themselves, nor inclusion relations between classes of them, incorporate a reflection of time. These set-theoretic world models can be intuitively seen as a collection of interrelated facts embodied in structured sets, a static picture, not a dynamical history. The 'computing' that is formalized in segment **a** and that physically happens in segment α is cast as a process of logical reasoning about the structures modeled in segment **c** through *S*-structures. In the 2370 year-old perspective of logic, computing and reasoning are very much the same thing. Logical reasoning proceeds by carrying out steps of logical inference, like in syllogistic arguments. The *premises* and the *conclusion* of each such an inference step are written down as symbolic logic expressions. The conclusion that comes out of executing one inference step become incorporated in the premises of the next step, leading to *inference chains*. Inference chains are stepwise transformations of symbolic configurations, a fact which establishes the connection to how-formalisms. A 'run' (or 'execution') of an algorithm can be understood as an inference chain which leads from an initial premise to a final conclusion. The initial premise is the *input* to the algorithm and the final conclusion is the *output*. These two symbolic configurations lie in the interface boundary **b**; they are the only items that are passed to and fro between the reasoning system **a** and the environment **c** that is being reasoned about. I return to the topic of time. Logical inference is not *temporal*. The relation between a premise and a conclusion is not that the former comes first in time and is temporally followed by the latter. Instead, the relation is semantical-implicational: if the former is true, then the latter is true too. Turing (1936) himself put this into plastic wording: *it is always possible for the computer to break off from his work, to go away and forget all about it, and later to come back and go on with it.*

By chaining, if the first input to an inference chain holds true in the set-theoretic world model **c**, then so does the ultimate output. This is the very same structure as of mathematical proofs in general, which likewise proceed from initial premises or axioms to the final claim in a series of argumentation steps. Hence, the parlance in theoretical computer science to call certain high-level algorithm formalisms *inference engines* or *theorem provers*. But, if inference steps are natively a-temporal, how then does time enter the picture? This is a natural question to ask, since after all human reasoning—the original inspiration for logic formalisms—evolves in real physical time. Any computational neuroscience that tries to understand human logical reasoning in terms of physical brain dynamics must provide an answer. But, in fact, this question is ignored in logic and computer science textbooks. Here is how I see it. In a dramatic abstraction, parts of reasoning in a physical brain can be regarded as a-temporal: namely, if it is assumed that a reasoning human can *store* symbolic configurations in un-alterable, one could say platonically immutable ways. An essential characteristic of symbols is that they just stay identically the same once 'written'. Symbols are what does not *change*; they are exempted from time. These two facts are two sides of the same coin: (i) that logic-based what-formalisms capture a-temporal logical inference relations, which become 'update' operations in how-formalisms, and (ii) that the native substrate of how- and what formalisms are symbol alphabets. When such how-formalisms are 'run' on physical computing systems, the physical hardware must necessarily provide for temporally unbounded, unaltering *memory* mechanisms where 'written' symbols defy the mutations of time. In physics terminology this means that digital hardware must incorporate subsystems that have timescales far longer than the use-time of the system.

This is also the right context to remark that by realizing logical theorem provers, digital computers can carry out anything that can be found in a mathematical proof. To the extent that some other, non-digital model of 'computing' can be mathematically formalized, it can be simulated by digital computers—except for the real-world temporal aspects of the simulatee. In this sense, the DC paradigm is and will remain the master of all others.

Take-home message 4: DC derives much of its superior powers from the very fact that semiconductor engineers have found ways to locally realize extremely large time constants, namely non-volatile memory devices wherein time virtually comes to standstill.

Hierarchical structuring of formal constructs. The formalisms used in DC for modeling 'computing' all admit to compose symbolic configurations into more compounded ones, which then can be used as building blocks in yet higher levels of *compositionality*. That human cognitive processing admits the creation of compositional entities is regarded as constitutional for human intelligence by proponents of classical AI and (Chomskian) linguistics.

Compositional hierarchies of formal constructs are present both in how- and what-formalisms. I discuss the former first. Virtually all programming languages (assembler languages possibly excepted) admit the definition of compound configurations (from 'lists' and 'arrays' to 'modules', 'objects', 'scripts') which bind together more elementary symbolic configurations into larger ones. Abstract models of 'computing' systems likewise all have provisions for defining compositional hierarchies, for instance by joining symbols into nested sequences on a Turing machine tape, by applying the lambda operator in lambda calculus, or by constructing parse trees in grammar-based formalisms. Compositional hierarchies of symbolic configurations in how-models correspond to an execution hierarchy of configuration update operations: 'executing' or 'evaluating' a higher-level, defined construct means to execute a sequence of lower-level update operations that happen inside the defined construct. The higher a compound formal construct in a compositional hierarchy, the larger the number of physical operations that are needed to realize the execution of the formal construct on a physical computing system. This multiplied physical effort leads to longer processing real time for higher-level construct execution—unless some *parallelization* scheme can delegate the required physical operations to a multiplicity of physical sites that operate simultaneously. Such time–space tradeoffs are an important topic in theoretical compute science. It is generally desirable, but not easy, to find formalisms and models of symbolic computing which lend themselves to high degrees of parallelization. In contrast to current models of symbolic computing, the human brain operates in an extremely parallel fashion. We see a face together with its nose, mouth and eyes. This is a lasting intellectual challenge for DC theorizing, and has motivated the title 'parallel distributed processing' for the bible book [Rumelhart and McClelland 1986] that marks the start of neural network research as we know it today.

A root mechanism in DC what-formalisms (formal logics) to capture the composition of cognitive entities is to construct nested *functional expressions*, as for instance TakeoffWeight(AirbusA320, Fuelfill(FlightLH237), NumberPassengers(FlightLH237, January24_2020)). In formal semantic theories of logic, such functional expressions refer to elements of the structured sets that make up *S*-structures. Such functional terms yield the root building blocks for syntactic expression-building in most logic formalisms. In a cognitive interpretation, such terms represent object concepts. Arbitrarily complex and varied syntactic composites can be built on top of them with other syntactic elements, like logical connectives AND, NOT, relation symbols, variables or extensive assertives like FOR ALL..., as in this first-order logic example: FOR ALL x: IF (x is Fruit AND Color(x) = Yellow AND Shape(x) = Curved) THEN x is Banana. All of these constructions are cognitively interpretable—they can even be (mis)taken for the essence of cognitive processing.

Take-home message 5: compositional formal structures are constitutive for DC models of computation. This makes DC how- and what-formalisms immediately suited to capture the (widely claimed but not uncontroversial) compositionality of human cognitive entities.

2.3.2. Probabilistic (sampling-based) formalisms and models

I now turn to probabilistic models of computation, marked with P in figure 3. As stated earlier, in my view an important criterion to call a physical system 'computing' is that its operations admit some kind of cognitive interpretation. For DC, this cognitive aspect is rational logical inference. In probabilistic 'computing' models the cognitive core aspect is the ability of animals and humans to make probabilistic inferences. These are formalized in probability formalisms through conditional probabilities of the kind, 'if the sky is cloudy, then the probability of rain is 0.3'. Stating and evaluating such conditional probabilities requires to have mental representations of *probability distributions*. I consider them the primary 'mental' or 'cognitive' objects in probabilistic reasoning and formalisms, analog to symbolic configurations in DC formalisms. Specifically, but not exclusively, the need for evaluating conditional probabilities arises when agents assess the chances that their actions will lead to the desired outcome. This view of biological cognition is one of the leading paradigms in cognitive science today and is referred to as *predictive brain* [Clark 2013], *free energy model of cognition* [Friston *et al* 2010], or the *Bayesian brain* [Tenenbaum *et al* 2006] hypothesis. In mathematics and machine learning we find a diversity of worked-out computational frameworks which formalize selected aspects of the predictive brain perspective. They include models of computing with probabilistic logic [von Neumann 1956], logic-oriented accounts of Bayesian statistics [Jaynes 2003], observable operator models and predictive state representations of probability distributions [Jaeger 2000, Littman *et al* 2001], Boltzmann machines [Ackley *et al* 1985, Hinton and Salakuthdinov 2006], reinforcement learning approaches to modeling intelligent agents [Basye *et al* 1995], or the neural engineering framework [Eliasmith *et al* 2012]. Here I restrict myself to formalisms and models which use *stochastic sampling* to represent and update distributions. In methods for sampling-based processing of probability distributions (SPPD for short), a probability distribution is not mathematically represented by a closed formula. Instead, it is approximately represented by a sample, that is a set of example points drawn from the distribution. Complex, high-dimensional distributions cannot in general be represented by analytical formulae. SPPD methods have become a major enabler for the simulation-based study of complex systems in physics [Metropolis *et al* 1953], for solving optimization problems, and in some branches of machine learning. They are often referred to as *Monte Carlo* or *Monte Carlo Markov Chain* [Neal 1993] or as *particle swarm* methods [Dellaert *et al* 1999]. The requisite random sampling processes still are mostly simulated on digital computers. SPPD techniques have however became also realized in non-digital physical computing systems, namely in DNA computing [van Noort *et al* 2002] for solving optimization and search problems, and more recently and with a wider application range in analog spiking neuromorphic hardware [Indiveri *et al* 2011, Haessig *et al* 2018, Moradi *et al* 2018, Neckar *et al* 2019, He *et al* 2019b]. According to the *neural sampling* [Buesing *et al* 2011, Pecevski *et al* 2011] view forwarded in theoretical neuroscience, temporal or spatial collectives of neuronal spike events can be interpreted (or used by the brain) as samples. Due to this inviting analogy to biological brains and the low-power characteristics of analog spiking neurochips, research in such hardware and corresponding 'algorithms' is expanding. But the main reason why I want to focus on sampling-based versions of probabilistic models is that they open views on 'computing' that differ from the DC views in interesting ways.

In order to preclude a false impression, I emphasize that there are many other ways to represent distributions besides via sampling. For instance, the procedural mechanics of certain (non-spiking, non-sampling) ANN models can be interpreted to compute parameters of distributions, or probability distributions can be approximately characterized and algorithmically processed through variational calculus. Both methods are widely used in current machine learning. Furthermore, firing sequences of neural spikes have been interpreted to encode 'information' in other ways than as delivering stochastic samples. Precise neural firing time patterns can be considered as carrying information in a variety of ways [Thorpe *et al* 2001, Izhikevich 2006, Deneve 2008].

Brushing over many specific differences between SPPD models, I will now give an account of their common traits and place them in the schematic of figure 3.

Formalism interrelations. I consider how-formalisms first. There are two kinds of how-formalisms and how-models in SPPD, depending on whether the targeted physical computing systems α are digital computers or not. In the first case, how-models are symbolic algorithms which, when executed, generate sequences of pseudo-random data points which accumulate over processing time into samples. Such algorithms are called *sampling algorithms* or just *samplers*. After down-compilation into low-level assembler programs these sampling algorithms can be passed to physical digital computers, crossing the modeling mirror. When the targeted hardware is non-digital, how-models describe the physical mechanics of the stochastic physical processes that are viewed as sampling processes. For biological brains, theoretical neuroscience offers a range of such models, from detailed physiological models of how spikes are created in neurons to more abstract castings like integrate-and-fire models of neurons or just Poisson spike trains. For analog spiking neuromorphic microchips, the sample-generating, physical stochastic processes in such microchips is modeled with electronic engineering formalisms on the device and circuit level [Indiveri *et al* 2011]. I would think that there also exist biochemical stochastic formalisms for modeling the reaction dynamics in DNA computing microreactors, but I am unfamiliar with that field.

The situation for all the sampling algorithms that have been proposed is markedly different from the situation in DC. In the DC domain, how-formalisms and models can all be precisely related to each other by mutual simulations. In SPPD I am not aware of a way how a stochastic sampling algorithm A could 'simulate' another such algorithm B. The representation of a distribution by a sample is inherently imprecise. The more data point examples are added to a sample, the more precisely it captures the distribution. Different sampling algorithms could possibly become related to each other by comparing their statistical efficiency (how many sample points are needed for a given accuracy level of representing a distribution). While this is an important optimization objective in practical SPPD designs [Neal 1993], I am not aware of a meta-theory or just attempts to systematically set different SPPD algorithms in relation to each other by comparing their statistical efficiency.

Turning to what-formalisms, these are the formalisms whose prime mathematical objects are probability distributions, considered as abstract objects independent of their concrete representation through samples or other formats. What-models are written down in the notation which students learn in probability or statistics courses. They gain their expressive powers mostly from stating and combining conditional probability relations. The analogy to the picture I drew for DC is obvious. The if-then format of conditional probability statements matches the if-then format of logical inference rules. In fact, according to one influential view of Bayesian statistics [Jaynes 2003], probabilistic what-formalisms can be considered and worked out along the guidelines of formal logic. There is however a noteworthy difference between DC and SPPD what-formalisms. In the DC domain, different logics can be related to each other through logic translations, yielding a (yet only partially explored) ordering along an expressiveness scale. Nothing like this can be found in probabilistic what-formalisms. There are no more or less 'expressive' formalisms of probability theory. It makes some sense to claim that there exists only a single probability what-formalism, namely the one that is taught in probability theory textbooks.

Take-home message 6: in both DC and SPPD domains, formal models for computing systems came as how- or what-formalisms. Unlike in DC, how-formalisms in SPPD are not related to each other, and cannot be transformed into each other in an obvious way, and one might also say that there exists only a single probabilistic what-formalism.

Intrinsic semantics. SPPD what-formalisms are expressed using the notation of mathematical probability theory. There are two major epistemological schools of thinking about 'probability', the *frequentist* (or *objectivist*) and the *subjectivist* (Bayesian) one. Their mathematical theorems mostly coincide in their surface format but are interpreted differently. In frequentist interpretations 'probability' means a physical property of real-world systems, namely a system's propensity to deliver varying measurement outcomes under repeated measurements, while in Bayesian interpretations, 'probability' means subjective degrees of belief. The frequentist account of probability directly arises from formal models of the physical environment (segment **c** in figure 3). These formal models of physical reality are a *probability spaces*, three-component mathematical structures standardly written as $(\Omega, \mathcal{F}, P)$, where Ω (called 'Universe' or 'population' among other namings) is a set of *elementary events* which can be intuitively understood as locations in spacetime where measurements could possibly be made; $\mathcal{F}$ is a certain set-theoretic structuring (a *sigma-field*) imposed on Ω; and P assigns objective probabilities to certain elements (called *events*) of this structuring. The formal connection between such world models $(\Omega, \mathcal{F}, P)$ and the descriptive what-formalisms that capture the probabilistic cognizing about the world (segment **a**) is established by *random variables* which create segment **b**. I remark that the term 'random variable' is entirely misleading. Mathematically, random variables are functions, not variables; and they are not random, but deterministic. To make this clearer: a statement like 'Peter is male' would be formalized in probability theory as $\mathsf{Gender}(\mathsf{Peter}) = \mathsf{Male}$, where Gender is the random variable, a function which deterministically returns the gender value from every concrete person (formally: from every element $\omega \in \Omega$). The randomness of random variables results not from that they somehow return random values, but that random arguments are given to them, following the probabilities prescribed by the item P in the world model $(\Omega, \mathcal{F}, P)$. The mathematics behind this is involved and I allow myself at this point to recommend my probability lecture notes [Jaeger 2019c] where I attempt to give an intuitive, yet rigorous introduction to the basic concepts of probabilistic and statistical modeling for readers who are not mathematicians.

Summing up the frequentist account of probability: the outside world **c** is cast as a structured set Ω of observation opportunities; observation apparatuses and procedures are abstracted into random variables; measured values (from ticked gender boxes in questionnaires to high-volume sensor data streams, filling segment **b**) become the data points of samples which are one way of formalizing probability distributions (segment **a**), which in turn can be seen as the key formal correlates of concepts in probabilistic accounts of cognition.

There are similarities and dissimilarities between the formalizations of intrinsic semantics in DC versus SPPD. The what-formalisms in both areas cast the physical world as highly structured sets (S-structures and sigma-fields) which represent, one might say, the preshaped substance of the modeled piece of the world. Both DC and SPPD what-formalisms are mathematically built around their respective intrinsic semantics. However, the meaning of 'meaning', that is, how some symbol or formula (in segment **a**) relates to something in the modeled world **c**, very much differs between DC and SPPD. Frequentist probability theory includes formal correspondents of measurement or observation procedures and their values as first-class citizens, namely random variables and the values that they can take. Segment **b** is central in probability theory, containing what is called *sample spaces*. Probability theory universally separates the measurable quality (like 'gender', 'speed') from the quantitative values that this measurement categories can take, like 'male' or '100 km h^{-1}'. Qualities become cast as random variables, quantities as the possible values of these variables. In contrast, the primary view adopted by logic formalisms identifies quality with quantity. That Peter is male would be formalized as $\mathsf{Male}(\mathsf{Peter})$. The symbol Male is a so-called *predicate* symbol, and predications (stating that certain objects have certain properties) are elementary operations in first-order logic. It is however also possible to express $\mathsf{Gender}(\mathsf{Peter}) = \mathsf{Male}$ in logic. To do this one has to introduce Gender as a *function* symbol. Still there is an important difference. In the logical understanding of $\mathsf{Gender}(\mathsf{Peter}) = \mathsf{Male}$, all three items (the argument Peter, the function Gender and the value Male) are contained in the S-structure, that is they are both located inside the world model in **c**. In a probabilistic interpretation of the same English statement, Peter is an element of the Universe Ω, sitting in segment **c**; Male is an observation value which is mathematically placed outside of the world model $(\Omega, \mathcal{F}, P)$ (I created the segment **b** to host it), and Gender links the two. This difference between logic and probability modeling grows from deep historical roots.

Take-home message 7: probability theory gives an account of how we observe the world; logic is about how we reason about an already observed world.

There is one more similarity between logic and probability modeling worth pointing out. Both SPPD and logic formalisms introduce separate symbols for every observable quality of the elements of the world-substance sets in segment **c**: random variables and function symbols, respectively (like Gender). An implicit assumption behind this symbolic naming of qualities is that the named quality is well-defined and *remains the same* throughout the time when the formalism unfolds in formal time (logic derivations or formalizing sampling in **a**), and/or when this computational process becomes physically instantiated (segment α), and/or as long as the real time evolves for the 'meant' interpretation in the real world environment γ or its model **c**. This is a highly nontrivial assumption and I will argue in section 3 that it bars the way to a generalized theory of computing.

Formal time is knitted into SPPD formalisms and models in intricate ways, often involving further theoretical elements reflecting space and temporal hierarchies. In physical computing systems (segment α), real time is consumed on a fast timescale to generate individual sample points, and on slower timescales to accumulate sample points into samples. In the view of neural sampling theories, where neural spikes (or possibly groupings of spikes) correspond to sample points, the fast timescale is the one of single spikes and the slow timescale the one of accumulating the effects of single spikes by mechanisms of temporal integration, for instance via building up neuron potentials or modulating synaptic efficiencies. Formal models of spike event generation (in segment **a**) can be expressed on different levels of granularity, ranging from modeling the electrophysiology of spike generation with differential equations to abstract representations of spike trains as Poisson processes. The accumulation of spike events into samples requires extensions of these formalisms to include some kind of temporal integration.—When running sampling-based algorithms on digital computers, the fast timescale reflects the runtime of program subroutines to generate sample points; these subroutines contain sub-subroutines to generate pseudo-random numbers which are typically encapsulated as primitives (the rand function) in programming languages. Generated sample points are formally accumulated in lists or arrays in program loops on the slow timescale of sample build-up.—This picture becomes more complex when sample points are generated in parallel strands, leading, among others, to *neural population* representations of samples in spiking neural network models, or to *Markov random field* models in non-neural algorithms (though the latter are mostly used for purposes other than sampling). But no degree of parallelization can entirely cancel the necessity for generating and accumulating sample points on respective fast and slower timescales.

Because the creation of samples needs time (formal or real, segment **a** or α), the cognitive core items in SPPD, namely distributions, are not defined for single moments of time or created instantaneously. A sample-based representation (segment **a**) or realization (in α) of a distribution becomes the more precisely defined the more sampling time is devoted to its generation. Sample-based representations of distributions are 'smeared' over time. If one adopts neural sampling views in cognitive neuroscience, this has nontrivial consequences for the notion of *mental states* which I will not further pursue here. The fact that samples develop over time leads to a spectrum of ways of how SPPD models of computation can be used for practical exploits, with different schemes for administering input. On one end of the spectrum we find scenarios where the computing system is 'clamped' to an unchanging input for the entire duration of the computation. Formally this means that some random variables of the model are frozen to fixed values. The sampling process is then run (in formal abstraction **a** or physical realization α) for an extended period of time, allowing the sampling to grow the sample large enough to decode from it the result with the desired degree of accuracy (I skip the complications of initial transients and ensuring ergodicity [Neal 1993]). Typical examples are classification tasks (for instance, input is an image, desired result is a probability distribution over possible classifications [Hinton *et al* 2006]) or optimization tasks (the archetype example: input is a roadmap, desired result is a round trip itinerary through all cities which with high probability is the shortest one [Kirkpatrick *et al* 1983]). On the other end of the spectrum, the input is itself temporal, for instance a sensor data stream, and the desired output is likewise temporal, for instance generated motor commands in a robot or a brain, or an estimation of an agent's motion relative to its visually perceived environment [Haessig *et al* 2018], or speech-to-text recognition tasks. This is the generic situation in adaptive online signal processing and control [Farhang-Boroujeny 1998] and in the study of *situated agents* [Steels and Brooks 1993], which comprise humans, animals, robots, software avatars or computer game characters. Solving such tasks, where input patterns or output patterns are themselves temporally evolving, requires sampling mechanisms where the next generated sample point depends on the history of previously generated ones. Important classes of formalisms of this kind include spiking recurrent neural networks (RNNs) [He *et al* 2019b], temporal restricted Boltzmann machines [Sutskever *et al* 2009] and sampling-based instantiations of dynamical Bayesian networks [Murphy 2002].—Intermediates between these two extreme ends of the spectrum (single fixed input versus nonstationary input streams) also occur, for instance when the input is a sequence of fixed patterns which are clamped each for some time before it is replaced by the next one.

Take-home message 8: the phenomenology of formal time in SPPD models is much richer than in DC. In DC, time reduces to discrete jumps from one well-defined, symbolic configuration the next, and these configurations are by themselves atemporal. In SPPD models, the core cognitively interpretable constructs, namely distributions, are themselves temporally defined through the sampling process. As a consequence, formal how-models must account for at least two timescales, and the formal interpretation of generated sample point sequences as distributions must account for nontrivial conditions like duration-precision tradeoffs or temporally overlapping realizations of different distributions. All of this is alien to the fundamental DC conception of executing algorithms.

Hierarchical structuring of formal constructs. Human cognitive processing admits—or in some views [Newell and Simon 1976], is even constituted by—the compounding, or 'chunking' [Newell 1990], of representational or procedural mental states or mechanisms into larger compositional items which then can again be composed again into even more comprehensive items, giving rise to compositional hierarchies of representations, mechanisms or processes. In DC this is accounted for by hierarchically organized symbolic configurations (in how-formalisms) and nested functional expressions in logic what-formalism. I can see three main ways how probability distributions can become hierarchically organized.

First, some how-formalisms explicitly arrange their random variables in layers, with 'low' layers modeling the sensor data periphery and 'high' layers modeling the cognitive interpretations of sensor input. This is the case for many instantiations of Boltzmann machines [Ackley *et al* 1985, Hinton and Salakuthdinov 2006]; furthermore, the conditional dependency graph of random variables in Bayesian networks and graphical models [Wainwright and Jordan 2003] can be hierarchically organized in reflection of a 'cognitive' stratification. These are generic probabilistic information processing formalisms. Task-specific layered spiking neural architectures, where deep feedforward neural networks are being re-coded into spiking neural substrates, have recently been receiving much interest, for instance in visual object recognition tasks [Yousefzadeh *et al* 2018]. Furthermore, I am aware of two complex information processing architectures which model complex human cognitive dynamics on the basis of spiking neural network substrates, namely Shastri's SHRUTI [Shastri 1999] series of connectionist models human reasoning and language processing and Eliasmith's almost-entire-human-brain models [Eliasmith *et al* 2012].

In these hierarchically or modularly structured systems, all random variables in all levels or modules are, in principle, sampled from with the same frequency. This is in agreement with the spiking dynamics in biological brains, which likewise has roughly the same timescale everywhere—neurons in the visual cortex do not fire orders of magnitude faster than neurons in the prefrontal cortex, although a hierarchy of processing levels lies between them. The 'cognitive' constructs, namely distributions determined through samples, are formally the same throughout levels or modules. Higher-level distributions are not made from, or composed of, lower-level ones. This is different from the compositional hierarchies in DC models, where higher-level symbolic configurations are made by binding lower-level ones into compounds. In the SPPD models pinpointed above, the assignment of being a 'higher' or 'lower' distribution is only in the eye of the human inventor of the model.

Second, hierarchies of distributions canonically arise in probabilistic models as conceived in Bayesian probability [Jaynes 2003, Jaeger 2019c], where distributions become themselves distributed in *hyperdistributions*. In the original motivation of Bayesian probability, these higher-level hyperdistributions reflect the subjective prior beliefs of an intelligent agent about which lower-level distributions are more or less plausible. Applying this principle to modeling probabilistic cognitive systems one obtains formalisms that are hierarchical in a substantial sense. The relationship between a distribution and a hyperdistribution is asymmetric. A hyperdistribution could be said to control, modulate or *bias* its lower-level children distributions. This gives rise to cognitive processing models whose dynamics unfolds in an interplay of bottom-up pathways (from sensor input to their cognitive interpretations) and top-down pathways (cognitive expectations modulating the perceptive filtering below). Prominent representatives of such bidirectional cognitive processing systems are Grossberg's adaptive resonance theory models [Grossberg 2013], Friston's free-energy models of processing hierarchies in brains [Friston 2005] and Tenenbaum's models of human cognition [Tenenbaum *et al* 2006]. However, although Friston's writings contain passing remarks on spiking neural dynamics, neither his nor other's models of Bayesian cognitive architectures appear to have been formulated or simulated (let alone physically realized) on the basis of sampling processes. Instead, when these authors (as most other neuro-cognitive modelers) want to capture temporal processing phenomena, they use formalisms that abstract from sampling processes by time-averaged neuronal firing rates governed by differential equations (Grossberg, Friston). Or, when their focus is on the a-temporal structure of hierarchies of (static) mental concepts, they adopt the formalism of Bayesian probability theory. The Bayesian distribution–hyperdistribution hierarchization should be analyzable in terms of sampling—meta-sampling hierarchies, at least if it is cognitively-biologically adequate and if brains actually use spikes for sampling. This would be a potentially rewarding mathematical project. Most likely this has already been done, but I am not aware of it.

Third, more elementary (lower-dimensional) distributions can always and precisely be mathematically combined into compound (higher-dimensional) distributions by *product* operations. Conversely, high-dimensional distributions can sometimes be approximately *factorized* into products of low-dimensional ones. Such factorizations are a major enabler to construct and evaluate high-dimensional probabilistic models of real-world systems in machine learning, computer graphics and physics [Huang and Darwiche 1994]. Factorization algorithms for distributions are being widely explored. Unfortunately, many real-world distributions cannot be satisfactorily factorized. A generalization of products of distributions is provided by *tensor product* representations, a standard operation in quantum mechanics [Coecke 2012] which also has been proposed as a paradigm for achieving analogs of symbolic compositionality in distributed neural activation patterns [Smolensky 1990]. However, the computational efficiency gains of factorizations are lost when the component distributions become *entangled* in tensor products. Like with Bayesian hierarchization, I am not aware of sampling-based accounts of product and tensor product operations, but again, this should be possible if such product operations are cognitively adequate and if the brain indeed exploits sampling for representing distributions.

Take-home message 9: compositionality of cognitive representations, mechanisms and processes—constitutive for symbolic computing—has currently no clear analog in sampling-based models of computing. When sampling is used as the core process for enacting 'computing' (as in Boltzmann machines or sampling-based evaluations of other graphical models), hierarchically organized interrelations between distributions exist only in the eye of the system designer but are not formally modeled on the level of interrelations between samples. To the extent that neural sampling is a valid view of information processing in biological brains, this means that either compositionality is not an inherent aspect of 'computing', or it means that mathematicians still have to work out—if they have not already—how hierarchies of distributions (distribution–hyperdistribution relations or factorizations of distributions) can be formally expressed through samples.

2.3.3. Dynamical systems (ODE-based) formalisms and models

I now turn to formalizations of 'computing' that are rooted in concepts and formalisms of dynamical systems theory, marked D in figure 3. Since almost a century, biological systems—neural and others—have been studied in a line of investigations which is referred to as *general systems theory* [von Bertalanffy 1968] or *cybernetics* [Wiener 1948]. This tradition co-evolved with the engineering science of signal processing and control [Wunsch 1985]. A landmark in interpreting the human brain as a dynamical (self-)control system is Ashby's classic *Design for a Brain* [Ashby 1952]. In another co-evolving strand, neural dynamics became modeled in a theoretical physics spirit, by isolating and abstracting dynamical neural phenomena into systems of differential equations, exemplified in the *Hodgkin–Huxley* model of a neuron [Hodgkin and Huxley 1952]. Later, when the mathematical theory of qualitative behavior of dynamical systems [Abraham and Shaw 1992] had matured and in particular after *chaos* and *self-organization* in dynamical systems became broadly studied, dynamical systems modeling rose to a commonly accepted perspective in cognitive psychology and cognitive science [Smith and Thelen 1993, van Gelder and Port 1995]. Today the separations between these historical traditions have almost dissolved. Mathematical tools from dynamical systems theory are ubiquitously employed in modeling neural and cognitive phenomena on all scales and abstraction levels, in a diversity that defies a survey. Even when seen only from within mathematics, dynamical systems theory is a highly diversified field. Its formalisms range from finite-state switching systems to field equations; time can be discrete or continuous; the state spaces on which dynamics are described can be finite or infinite sets, vector spaces, manifolds, function spaces, graphs or abstract topological spaces, etc. Furthermore, this mathematical field has overlaps with statistical physics, information theory, stochastic processes, signal processing and control, game theory, quantum mechanics and many more.

Here I will only consider models expressed with ODEs which have the familiar look $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a})$, where $\mathbf{x} \in \mathbb{R}^n$ is an n-dimensional real-valued state vector, $t \in \mathbb{R}$ captures continuous time, the dot notation $\dot{\mathbf{x}}$ is a shorthand for the first temporal derivative $\mathrm{d}\mathbf{x}/\mathrm{d}t$ and the optional terms $\mathbf{u}(t)$ and $\mathbf{a}$ denote input signals and control parameters. The function f defines a *vector field* on the *state space* $\mathbb{R}^n$ which indicates the local direction and velocity of state motion. Higher-order ODEs which contain higher-order derivatives like $\mathrm{d}^n x/\mathrm{d}t^n$ can be transformed to first-order ODEs by introducing auxiliary variables. Such ODE models are by far the most widely used and most deeply studied kind of dynamical systems models; they are what mathematics students find in introductory textbooks like Strogatz (1994). The mathematical theory of qualitative behavior in dynamical systems—attractors, bifurcations, chaos, modes, 'self-organization', etc—has been first and foremostly been developed for $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{a})$ system models. ODEs is the most broadly used kind of formalism in the modeling of continuous-time dynamics in biological and ANNs and cognitive processing; and it is the formalism used by electrical engineers when they design analog neuromorphic circuits [Indiveri *et al* 2011].

Of course *partial differential equations* (PDEs) are important too. They are needed for modeling spatiotemporal dynamics on continuously extended substrates, for instance in neural fields [Lins and Schöner 2014], in non-Boolean spin-wave devices [Csaba *et al* 2017], in analog computers that exploit the spatial nonlinear dynamics in macroscopic sheets of materials [Mills 2008], or reaction–diffusion computers [Adamatzky *et al* 2005]. The spatial organization of physical computing systems is obviously important to design and analyze them. Yet I have deliberately left out this aspect from this study (besides other important aspects, see section 2.2) in order to not make this article even longer than it already is. A halfway appropriate discussion of spatial organization would have to consider not only PDEs, which are the natural tool to characterize dynamical phenomena in homogeneous material substrates, but also many other mathematical tools (for instance cellular automata, cellular neural networks, finite-element methods, Petri nets, dynamics on networks, dynamics on topological spaces) most of which are better suited than PDEs when it comes to dynamics on inhomogeneous substrates or wire-connected discrete devices. Many of these modeling techniques however admit approximations through ODEs. For instance, PDE models can be approximated by ODEs with the tools of finite-element methods. Since furthermore the theory of ODEs is the most classical and the deepest worked-out one among all of these others, and since ODEs is the standard formalism in most work in NC, I opted for them on this occasion.

Formalism interrelations. When discussing DC and SPPD, I found it useful to distinguish how- from what-formalisms. I will follow this strategy again and discuss how- and what-formalisms, starting with the former.

In ODE modeling we find only a single how-formalism. It is the canonical textbook ODE formalism. Using its notation, a researcher or engineer can write down how-models $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a})$ (in segment **a**) which describe physical computing systems (in α). These models directly capture the real-time, metric dynamics of continuous variables in computing systems—voltages and currents in analog microchips and neuronal circuits, 'activations' of neural assemblies or concepts in models of neural cognitive processes. While there are inclusion hierarchies of classes of such ODEs—foremostly, ordering them by the highest order of derivatives, speaking of first-order, second-order etc ODEs—the ODE formalism per se is always the same. This is different from the situation in DC where different formalisms (like machine models, programming languages) are related to each other by simulation (encoding, compilation) relationships. Such inter-formalism translations play no role in ODE modeling.

This is however true only if we restrict the discussion to ODE modeling. If we would include other dynamical systems formalisms into our discussion, numerous mutual translation or simulation relations can be found. While in the DC domain such simulations are exact, in the dynamical systems world they typically entail approximations. In particular, PDE models can be approximated by ODEs through finite-element modeling, and continuous-time, continuous-space formalisms (ODEs, PDEs, field equations) can be arbitrarily well approximated by discrete-time, discrete-space formalisms (like numerical ODE solvers or cellular automata). Unlike in DC, where such formalism translations are student homework exercises, finding such continuous-to-discrete encodings in the dynamics domain is not trivial and analyzing the approximation quality can be demanding.

What-formalisms and models in the ODE context are more complex and subtle. I will have to spend some paragraphs to explain how I see them.

What-models capture *qualitative* phenomena in dynamical systems. Such phenomena appear when state trajectories $(\mathbf{x}(t))_{t\in\mathbb{R}}$ fold into the state space $\mathbb{R}^n$, giving rise to *phase portraits* which are populated by a zoo of *geometrically defined* objects like point attractors or repellors or saddle-nodes, periodic orbits, chaotic attractors, stable and unstable manifolds, basins of attraction etc. The fascinating world of qualitative phenomena is most beautifully revealed in the copious picture-book of Ralph Abraham (pioneer of dynamical systems theory) and Christopher Shaw (graphics artist) which I enthusiastically recommend as first reading for novices in dynamical systems [Abraham and Shaw 1992]. Qualitative phenomena are preserved when the state space is smoothly distorted by 'rubber-sheet' transformations: if one draws a closed loop on the surface on a slack balloon and then stretches or inflates it, the drawn figure will remain a closed loop.

When (neural) dynamical systems are used to model cognitive processes, it is these geometrical objects which are stipulated as correlates of cognitive phenomena. The most widely explored correspondence of this kind is to consider cognitive *concepts* as represented by *attractors*. This is a richly worked-out theme with a long history. It can be traced back at least to the notion of cell assemblies formulated by Hebb (1949), reached a first culmination in the formal analysis of associative memories [Palm 1980, Hopfield 1982, Amit *et al* 1985], and has since then diversified into a range of increasingly complex models of interacting (partial) neural attractors and associated 'attractor-like' phenomena [Yao and Freeman 1990, Tsuda 2001, Rabinovich *et al* 2008, Sussillo and Barak 2013, Schöner 2019]. But other geometrical objects have also been selected as representatives of concepts, for instance solitons and waves [Lins and Schöner 2014] or algebraically defined regions in state space [Tino *et al* 1998, Jaeger 2017], and many other cognitive entities besides concepts have been modeled by qualitative phenomena in dynamical systems [van Gelder and Port 1995, Besold *et al* 2017]. Because I require interpretability in cognitive terms from what-models, the domain of qualitative-geometrical phenomena is the natural candidate to look for what-formalisms, models and theories in ODE systems.

Unfortunately there is no canonical mathematical formalism or theory to comprehensively describe the world of geometric wonders that arise in general ODE systems. Only for special classes of dynamical systems, and mostly confined to low-dimensional ones, elementary qualitative phenomena and their geometrical relations to each other have been accessible to theoretical analyses, in particular in the context of Morse–Smale theory [Shub 2007]. Otherwise, mathematicians invariably use plain English and computer graphics besides formulas to describe how attractors and other geometrical objects become related to each other. New kinds of qualitative phenomena are continually being identified. Most insights into the geometry of dynamics that are today available have been discovered, defined and studied for *autonomous* dynamical systems only. These are systems whose equations $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{a})$ have no input term. The study of qualitative behavior in input-driven systems, which have an additional input term $\mathbf{u}(t)$ in their ODEs, is in its infancy [Kloeden and Rasmussen 2011, Manjunath and Jaeger 2014]. Current mathematical theory offers only painfully limited ways of inferring which qualitative phenomena will emerge from a given how-model $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{a})$ or $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a})$. This has to be worked out on a case-by-case basis. Analytical understanding is often impossible to achieve, and numerical, intuition-guided simulations on low-dimensional subspace projections afford the only window of insight.

Thus I cannot point to existing worked-out what-formalisms for ODE modeling. It might be well the case that a comprehensive what-formalism, and what-theory expressed on its basis, is principally impossible for ODE systems. Just like natural scientists will forever continue to discover new qualitative phenomena in nature, and just like biological evolution incessantly finds and exploits new qualitative phenomena on a 'keep *any*thing that works' basis, mathematicians may be in for a principally open-ended discovery journey. This open-endedness may be intrinsic in an ill-definedness of what *qualitative* means. Just like biological evolution discovers ever more qualitatively new 'tricks' to enhance the powers of brains, where every newly found trick may fall out of the reach of a previously sufficient theory, GC theorists may find themselves always lagging behind inventive GC system engineers who find ever more ways to harness physical-dynamical phenomena for computing.

This is disconcerting. Physicists rely heavily on ODEs and other kinds of differential equations to model physical systems, and so do electrical engineers when they design circuits for analog microchips. ODE formalisms thus seem an inviting platform for developing a theory for GC which is naturally able to connect to the physics of computing systems. But the difficulties and possible open-endedness of formulating qualitative what-theories are a serious obstacle. I see four options to deal with this roadblock:

(a) Accept that a closed, complete what-theory is not possible and embark on GC theory building in the spirit of an open-ended discovery expedition, fencing the bestiary of qualitative phenomena with a forever expanding zoo of what-theories.

(b) Build a GC theory around an *a priori* restricted set of qualitative phenomena. This is what DC does in relying, deep down, only on the phenomenon of bistability which gives the 0–1 bits that symbolic computing is made of. This is also what we see in most neural network models in computational neuroscience and machine learning, when for instance a specific class of attractor-related phenomena is chosen to discuss aspects of cognitive processes. The obvious drawback is a limitation of perspective which will miss the majority of potentially relevant qualitative phenomena as well as the majority of aspects of cognitive processing.

(c) Not adopt a dynamical systems oriented perspective. This can be utterly successful as witnessed by DC, but I am afraid it bars the way to exploiting most of what physics can offer for 'computing'.

(d) Find a new foundational mathematical formalism for dynamical systems which makes qualities, not quantities, the objects of dynamical change. This is what I find the most promising route, and I will put forth initial ideas in the conclusion section.

In summary, the time is not ripe yet for a systematic discussion of what-formalisms, models and theories for ODE models or dynamical systems models in general—but it can be seen that here is a challenge that deserves to be addressed.

Take-home message 10: in dynamical systems oriented modeling of computing systems (restricted here to ODE modeling), the single how-formalism is the textbook ODE formalism. In computing systems modeled by ODEs, a plethora of geometrical, qualitative phenomena may arise. They are candidate formal objects to be interpreted in what-models as correlates of 'cognitive' phenomena. A few of them, in particular attractors, are already widely being perceived in the cognitive and neurosciences. A systematic organization of such studies in a compendium of interrelated what-theories remains a task for the future.

Intrinsic semantics. In L and P modeling there are canonical ways how the environment (γ in figure 3) is mathematically modeled (in segment **c**), namely by the set-theoretic constructs of *S*-structures and probability spaces. In dynamical systems oriented modeling no canonical view on how to model the environment exists. The question is rarely asked and I dare say under-researched. There are historical reasons for this semantic almost-blindness. Starting with Newton, dynamical systems modeling has for a long time been the homeground of physics. Experimental physicists try hard to *isolate* their system of interest from the environment. This made them use a kind of ODE which are, in mathematical terminology, *autonomous*, that is, their equations $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{a})$ have no input term. The historical development of dynamical systems mathematics was by and large confined to autonomous systems. Starting, say, in the 1960s, there was explosive mathematical progress in perceiving, analyzing and (importantly) visualizing [Peitgen 1986] qualitative phenomena like attractors or chaos. Other disciplines besides physics and even the general public became fascinated by dynamical systems and began to interpret their respective objects of study as qualitative dynamical phenomena. In the cognitive sciences this became an established perspective around the year 1990 [Schöner *et al* 1986, Smith and Thelen 1993, Port and van Gelder 1995, Jaeger 1996]. However, the available mathematical tools and metaphors were largely confined to autonomous input-free systems. This barred the way to develop dedicatedly semantic accounts of how the modeled systems interact with their environment. In neuro- or cognitive modeling, qualitative phenomena inside the modeled neural or cognitive system were not related to its outside environment, but were mapped to system-internal cognitive constructs and processes. An important theme was (and is) how symbols and 'concepts' can be interpreted by qualitative dynamical phenomena—the neuro-symbolic integration problem [Besold *et al* 2017].

In my opinion, in order to work out a genuine intrinsic semantics for dynamical systems models of 'computing' systems, one would have to move from autonomous system models like $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{a})$ to input-driven ones like $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{a})$. But, as I remarked earlier, mathematical theory development for qualitative phenomena in such *non-autonomous* systems is far less developed than for autonomous ones [Kloeden and Pötzsche 2013]. Even clarifying the notion of an attractor—the centrally focused dynamical phenomenon in current dynamical systems oriented cognitive modeling—is loaded with mathematical intricacies [Poetzsche 2011, Manjunath and Jaeger 2014]. It will take a while before this venue can be widely explored outside an inner circle of specialized mathematicians.

But of course, agent-environment interactions have been broadly studied from dynamical systems oriented perspectives. I highlight examples that I am familiar with.

Two classical fundamental models of embedded living systems are von Neumann's model of self-replicating automata [von Neumann and Burks 1966] (which also marks the invention of cellular automata), and the model of *autopoietic* living systems by Varela and Maturana [Varela *et al* 1974]. Although both models are not formulated with ODEs but via discrete-time, finite-state cellular automata, I mention them here due to their dynamical systems look-and-feel. In both cases, the 'computing', rather: 'living' system is modeled as a delimited, moving, shape-changing and growing area within a cellular space–time grid.

The theory of autopoietic systems is even explicitly anti-semantic. According to their originators, what happens inside such a system is not reflecting, or representing, outside givens: '*In this sense we will always find that one cannot understand the phenomenon of cognition as if there were "facts" and objects "out there", which one only would have to fetch and put into the head.*' (my translation from a German translation [Maturana and Varela 1984] of Maturana and Varela's book *El árbol del concocimiento*). Instead, an autopoietic system constructs its own internal world, which is shaped and connected to the outside only because this organization has to meet requirements of self-stabilization and survival. Maturana and Varela have coined the term *structural coupling* for this principle [Maturana and Varela 1984]. Their biologically motivated theory has given rise to schools of epistemology called (radical) *constructivism* [Schmidt 1987] and *enactivism* [Wilson and Foglia 2017]. As far as I am aware, a mathematical formalization of structural coupling in terms of qualitative phenomena in dynamical systems has not been attempted.

Ashby's 'Design for a Brain' [Ashby 1952] proposes a fundamental explanation of the brain's function and purpose in the tradition of cybernetics. Modeling both the brain and its environment through differential equations, he focuses on mechanisms of homeostatic self-regulation and adaptation to changing environmental conditions. In an extensive methodological discussion (chapter 3) he defends the mathematical approach to model both the brain and its environment by coupled differential equations.

Beer's model of a situated (insect-level) agent [Beer 1995], formulated much later than Ashby's model but in the same spirit and explicitly referring back to it, engages in the scientific controversies of the time where *behavior-based robotics* [Brooks 1989] or *new AI* [Pfeifer and Scheier 1999] developed in opposition to traditional, logic-based, symbolic AI. In a long section, Beer (1995) refutes the idea to call a brain 'computing' (at least if understood in the Turing/DC sense), rejecting the symbol-referent semantic paradigm, and renews Ashby's approach of modeling the agent in its environment as two coupled dynamical systems, each modeled by ODEs. Using mathematical concepts and simulation technology that were not available to Ashby, Beer explores how geometric structures in the phase portrait of the agent are influenced/shaped in response to geometric information obtained from the phase portrait of the environment. In a nutshell, he argues in favor of replacing the semantic reference relation by a dynamical interaction relation between two equal partners.

In nonlinear control theory we find an analog of the agent-environment relation, namely between a *controller* and the controlled *plant*. This analogy is in fact directly instantiated in the machine learning field of reinforcement learning, which on the one hand is usually presented as an agent acting in an environment and on the other hand is a special sort of an *optimal control* problem, with substantial cross-fertilization between the two fields. However, the optimal control setting requires a probabilistic treatment which would lead out of the present context.

Controller-plant systems can often be formalized as two coupled ODE subsystems. Call the state vectors of the controller and the plant $\mathbf{x}_\mathrm{c}$ and $\mathbf{x}_\mathrm{p}$, respectively. The controller can monitor the plant through *observations* $\mathcal{O}(\mathbf{x}_\mathrm{p})$, and the plant is influenced by *controls* $\mathcal{C}(\mathbf{x}_\mathrm{c})$. This leads to coupled equations $\dot{\mathbf{x}}_\mathrm{c} = f_\mathrm{c}(\mathbf{x}_\mathrm{c}, \mathcal{O}(\mathbf{x}_\mathrm{p}))$, $\dot{\mathbf{x}}_\mathrm{p} = f_\mathrm{p}(\mathbf{x}_\mathrm{p}, \mathcal{C}(\mathbf{x}_\mathrm{c}))$, which is a special format of two coupled ODE systems. The controller furthermore typically receives an external *target* input, which I omit here (it would represent the user of the system). This simple controller–plant schema is a stark simplification; practical designs usually include several more modules serving specific subtasks [Jordan 1996]. In control engineering and also in the motor control research in neuroscience and robotic, the focus of modeling lies on understanding conditions for dynamical *stability* of the controller–plant system, which naturally leads to analyzing convergence conditions to attractors. There are two ways how attractors become important in control theory. First, they can express the mathematical condition that the coupled controller–plant system behaves in a stable way, being robust against perturbations. They are then a characteristic of the dynamical behavior of the coupled system in its joint state space $\mathbf{x}_\mathrm{c} \otimes \mathbf{x}_\mathrm{p}$ and not a carrier of semantic relations. Second, the controller may become subdivided into a path planner and an execution controller. While the planner module is mostly formalized with mathematical tools other than ODEs, sometimes ODEs are used also for this module. The attractors are then located inside the planner and represent objects or conditions in the external task space of the plant, for instance target positions or paths for a reaching motion [Bizzi and Mussa-Ivaldi 1995, Khansari-Zadeh and Billard 2011]. This would be in line with the view that I advocated for ODE what-formalisms above, namely that geometric objects in the phase portrait should be cognitively interpretable and amenable to a semantic interpretation. I am however not aware of work aiming at a formalization the relationships between attractors in control planning modules and external givens.

In my understanding of the term, for a relationship between two formally modeled domains to be called ‘semantic’ it is necessary that there is a difference in kind between the two domains ((**a**) and (**c**) in figure 3). The models (**a**) of brains, agents, controllers—that is, GC systems—should contain formal elements that can be understood as forming *representations* of some sort: descriptors, percepts, words, concepts, formulas, symbols, statements, etc which can be formally interpreted as referring to *referents* in the external world domain (**c**): objects, situations, materials, events, etc. In dynamical systems oriented modeling, one typically finds that these are made of the same mathematical substance as the models of their environment—often ODEs, and semantics collapses to interaction between two subsystems of a single dynamical system.

Take-home message 11: the study of how cognitively interpretable, qualitative phenomena arising in the internal dynamics of an agent relate to dynamical models of its environment has barely begun for ODE formalisms. Appropriate ways of achieving this may have to wait for a further development of non-autonomous dynamical systems theory. Existing dynamical models of agent–environment interaction use the same formalism for both, namely dynamical (sub)systems described with the same formalism. While this helps to understand living or artificial autonomous agents and stability or viability conditions, it does not express or explain the agent-internal processes as ‘computing’.

Formal time. The formal time model is hidden in the dot in $\dot{\mathbf{x}}$, the shorthand for the derivative $\mathrm{d}\mathbf{x}/\mathrm{d}t$ with respect to time t. By inserting *time constants* τ in the governing equations $\tau\,\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a})$, the system evolution can be proportionally sped up or slowed down by making τ smaller or larger.

When a natural scientist or engineer sees an equation $\tau\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a})$, without further thinking he or she will understand the variable t to stand for the continuous arrow of real-world physical time, one of the reasons why ODE formalisms are so natural for physicists and engineers. This is fundamentally different from the formal time in DC modeling, which is logical-inferential in its origin and becomes mapped to the physical time of physical computers in essentially arbitrary scalings, depending on the clock frequency of the digital processor. One will not find the word ‘second’ mentioned in any textbook of theoretical computer science! The fact that a formal ‘1 s’ segment of t really means one physical second in ODE analog circuit models has important consequences for analog computing practice. First, the designer of analog circuits must match time constants in his/her formal ODE models to the physical time on board of the microchip. Second, assigning different time constants to different system variables is an utterly delicate affair, because slightly different settings of these time constants may lead to qualitatively different system dynamics. Third, analog computing systems must be timescale-matched to their physical task environment because both operate in the same physical time. On the positive side, this makes it natural to design analog computing systems that are physically embedded in their environment through continuous input- and output signals. DC faces deep-rooted difficulties with continuous real-time processing and the common solution there is to build or buy machines whose digital clock frequency is so fast that the inevitable delays of ‘processing’ the current input value become non-disruptive for the task at hand. It is however technically difficult to construct analog hardware that is both small-sized and slow. In analog electronics, small-sizedness conflicts with slowness because slow time constants spell out into large capacitors or very low voltages, both of which quickly hit limits. In our own work with analog neuromorphic microchips we found this the most challenging obstacle when we wanted to realize an online heartbeat monitoring system—human heartbeats is much slower than time constants in nanoscale electronic circuits [He *et al* 2019b].

These difficulties render the practice of designing and using analog (especially neuromorphic) computers in ‘situated’ online processing tasks markedly different from how digital machinery is used. One cannot use the same machine and neural network model for fast and for slow tasks. The formal model timescales must match the physical machine's which in turn must match the ones of the task. In DC, the only limit regarding timing in situated online processing scenarios is given by the digital clock frequency. Provided that an adequate 'real-time' operating system is available, any task with slower timescales can be solved with digital machines by buffering variable values for waiting times as long as the slow task timescale demands. Furthermore, the qualitative behavior of analog circuitry and their how-models depends sensitively on the settings of numerical parameters in the models, especially (but not exclusively) on time constants. On the plus side: analog computing systems that have been successfully embedded in their task environment with matched time constants and internal processing *entrained* by the driving input can cope with broad-band signal interfacing with virtually zero delay.

This picture of online-entrained operations of analog computing circuits corresponds to reflexive, automated operation modes of biological brains, of particular importance in motor control. But brains of not all-too-primitive animals also command *working memory* and *long-term memory* mechanisms where information packets can be encoded and stored for extended durations. Biological brains apparently use a variety of mechanisms for such extended-time memory functionalities [Durstewitz *et al* 2000, Kistler and de Zeeuw 2002, Baddeley 2003, Tetzlaff *et al* 2012, Preston and Eichenbaum 2013, Fusi and Wang 2016]. These mechanisms allow biological brains to integrate information input over timespans that are much longer than the *transient* timescales given by time constants in ODEs, and realize processing procedures that are akin to DC procedures. Formalizing and analyzing mechanisms of encoding, storing, addressing, accessing and decoding information packets lies beyond the possibilities of pure ODE models and requires additional descriptions of multi-module architectures, encoding principles and learning processes. These additional descriptive components are typically provided in plain English or graphical flow diagrams [O'Reilly *et al* 1999, Howard 2009, Markert *et al* 2009, Pascanu and Jaeger 2011] since there exists no formal theory yet which captures information encoding, storing and retrieving in continuous-time dynamical systems.

Take-home message 12: in ODE-based how-models the timescales of the model must match the physical timescales of the I/O signals if the physical computing system is used in online tasks. This is similar to the operating conditions of biological brains when they are entrained in online processing tasks, but very different from DC. However, ODE models can be used as modules in complex memory architectures which mirror structural and procedural principles known from symbolic computing. In the absence of a general theory of persistent information encoding in ODE models, such architectures are informally described on a case-by-case basis.

The **hierarchical structuring of formal constructs** is intimately connected with timescales in ODE modeling. When researchers in cognitive neuroscience, robotics and autonomous agents or machine learning conceive of their respective intelligent agent architectures as dynamical systems, they almost by reflex assign *fast timescales* to subprocesses that operate close to the sensory–motor interface boundary (segments **b**, β), and the assign *slow timescales* to subprocesses operating at 'higher' levels of the cognitive processing hierarchy. In my opinion, timescale hierarchies are key for finding hierarchicity in dynamical systems.

Compared to the state of the art in characterizing geometry-defined qualitative phenomena, the study of timescale-defined qualitative phenomena is in an early stage. Most qualitative phenomena arising in ODE systems are described via static geometric-topological conditions in phase portraits. Time is by and large factored out in the definitions of phenomena like fixed points, attractors or bifurcations. The 'speed' of state variable evolution is irrelevant for the mathematical definition of these constructs. Time enters these definition at best in the form of asymptotic (infinite duration) characterizations of convergence.

Multiple timescale dynamics are of general interest for *complex systems* modeling in all natural sciences. A rich body of mathematical theory has grown [Kuehn 2015]. When writing down ODEs, slow versus fast timescales can be imposed on system variables by multiplying time constants into their differential equation. But as we have seen in the previous subsection, this freedom is not granted to hardware or brain modelers because the time constants in their models must match a given physical reality. However, in ODE and other dynamical systems all of whose variables are fast, new descriptive variables can be discovered through mathematical analyses which reflect slowly changing *collective* characteristics of the underlying system. The most common strategy is to use *mean-field* methods where many system variables are averaged, but it can be done in many other ways. For instance, complex chaotic attractors can be described in terms of their make-up from connected lobes, which can be discerned from each other geometrically (by approximating them with periodic attractors) or by registering dwelling times (the dynamics stays for some time in one lobe before it moves to another) or by the possibility to stabilize the lobes [Babloyantz and Lourenço 1994]. A method developed in computational neuroscience and machine learning is *slow feature analysis* [Wiskott and Sejnowski 2002] where slowness of derived variables is directly used as criterion to identify them. In physical systems, the emergence of multiple timescales is often connected to mechanisms that cannot be captured with ODEs, in particular delays in system-internal signal propagation and spatial extension of systems.

Mathematical and applied research on multiple timescale dynamics in ODE models is so rich that a comprehensive treatment cannot be attempted here. I single out one kind of system model which offers relevant insights for understanding ODE systems as ‘computing’. These are RNN models where a stack of neuronal layers corresponds to a sequence of increasingly slower timescales, starting from a ‘low’ fast sensor–motor interface layer to ‘higher’ layers whose increasingly slower neuronal state variables are interpreted as representing increasingly compounded ‘features’ or ‘concepts’. Examples are RNN based robot control systems where these timescale differentiations are predesigned into the neuronal hierarchy through time constants [Yamashita and Tani 2008]; or a hierarchical RNN model for the unsupervised discovery learning of increasingly slower components in multiple-timescale input signals, where the different timescales are predesigned by way of using faster or slower ‘leaking’ in the used neuron model [Jaeger 2007]. In neuroscience, the idea that biological brains operate at fast timescales close to their sensor–motor periphery, and at increasingly slower timescales as one moves ‘upward’ in the anatomical hierarchy toward forebrain structures, has been advanced on the basis of theoretical argument corroborated by simulation studies [Kiebel *et al* 2008], and it also has been confirmed by physiological measurement [Murray *et al* 2014]. In machine learning architectures the formal neurons in higher layers are made slow by adorning their ODEs with slower time constants—‘higher’ neurons are slower neurons. With regards to biological brains, it is still unclear to which extent (and on the basis of what physiological mechanisms) individual neurons in different brain regions are slower or faster; or how or to what extent different timescales of *neural integration* of information arises from collective mechanisms [Goldman *et al* 2008]. Even when all neurons across the hierarchy are individually equally fast, slowness in higher layers can arise as a property of derived variables like spiketrain autocorrelation timescales [Murray *et al* 2014] or collective variables like local field potentials [Kiebel *et al* 2008]. The low-to-high layering in hierarchical RNN architectures can be *feedforward* (lower layers feed only to higher layers but not vice versa) or *bidirectional* (with both ‘bottom-up’ and ‘top-down’ couplings between the layers). Biological brains are eminently bidirectional, as are many formal models in computational neuroscience and a few in machine learning. I will only consider bidirectional models here. With regards to timescales there are noteworthy similarities and dissimilarities between bidirectional hierarchical RNN models and digital computer programs. Large computer programs are written by professional programmers in a hierarchically organized fashion, with higher-level program modules (scripts, functions, objects) calling lower-level ones as subroutines. Such programs are bidirectional: the higher-level module ‘calls’ the lower-level one and typically sends down initialization or configuration parameters; and the lower-level subroutines send the results of their computation upwards to the calling module. This automatically leads to a runtime hierarchy: executing higher-level modules necessarily takes longer than executing the lower-level because the latter are called inside the higher-level one. The resulting timescale hierarchy could be called a waiting-time hierarchy: the symbolic configuration update step assigned to the higher-level module is pending in an indeterminate status as long as the called lower-level subroutines are being executed. In contrast, in bidirectional RNN systems, all neurons on all levels are active simultaneously, and the top-down and bottom-up information flows are continually streaming without need (or opportunity) for waiting. This is yet another reason why brains cannot easily be likened to symbolic computing machines, and why timescales are a key issue when one wants to develop conceptualizations of ‘computing’ that extend beyond the DC paradigm and include brains.

For mathematical analyses of top-down influences from higher to lower layers it is decisive *how much* slower the higher layer develops. If the timescale separation is very large (say two orders of magnitude or more), the slow evolution speed in the higher layer can be approximately regarded as standing still compared to the fast lower layer. It is then widespread but not entirely rigorous practice to consider the variables fed to the lower from the higher layer as constant control parameters in the equations governing the lower layer, and study different processing modes of the lower layer separately by considering different static settings of these control parameters. A rigorous mathematical analysis of such *slow–fast systems* needs the tools of singular perturbation theory [Kuehn 2015]. If the timescale separation is not very large, analyses become difficult and rely on heuristic assumptions, as in the proposal of Wernecke *et al* (2018) where the emergence and decay of transient qualitative phenomena in the lower fast layer is regarded as being guided by the attractor structure that would be defined if the timescales were widely separated.

Strictly layered architectures are popular in machine learning, AI and control engineering [Albus 1993]. When they are bidirectional, they contest the limits of today’s mathematical analyses. Biological brains [Felleman and Van Essen 1991] and complex practical control architectures [Thrun *et al* 2006] have a modular structure which is more complex than a linear ordering of processing layers, with many lateral and diagonal processing pathways. Furthermore, it may happen that a variable that is fast at some time may turn into a slow one later and vice versa. Analyzing, or merely characterizing such timescale meta-dynamics is beyond the reach of today’s mathematics as far as I can see.

How does all of this relate to my distinction between how- and what-models? An answer is not easy because timescales are investigated and formalized in many ways in different contexts. I would say that timescale hierarchies are relevant both for how- and what-modeling, but unfold their full power especially in what-modeling. When electrical engineers write down their device or circuit specifications in ODEs, adjusting their time constants to their target systems, they are doing a how-modeling job. But in doing this they will not be considering the cognitive-representational implications of what will happen when their systems start 'computing'. Computing theorists on the other hand will consider timescales a qualitative property that admits a cognitive interpretation. Then these models appear as what-models. This view is immanent in the intuitions about physical intrinsic semantics held by cognitive modelers who posit that brain-internal variables represent objects or situations in the external environment, as revealed in this quote: '*Many aspects of brain function can be understood in terms of a hierarchy of temporal scales at which representations of the environment evolve. The lowest level of this hierarchy corresponds to fast fluctuations associated with sensory processing, whereas the highest levels encode slow contextual changes in the environment, under which faster representations unfold.*' [Kiebel *et al* 2008].

Take-home message 13: timescale hierarchies, possibly structurally reflected in layered architecture designs, are an important ingredient of dynamical systems models to qualify them as 'computing'. Different timescales in (neuromorphic) analog hardware and their mathematical models can arise from a variety of formal and physical effects for which we still lack a comprehensive understanding. The formal nature of multiple timescales differs fundamentally between digital computers (waiting time hierarchies) and parallel analog systems (continually interpenetrating bottom-up and top-down streams of information with different time constants).

2.4. Section summary

The purpose of this section was to help myself and the reader to understand better what we may mean when we speak of a 'theory of computing'. This is a fairly well demarcated concept in symbolic/DC—there is a choice of university textbooks for 'Theoretical Computer Science' which by and large all present the same canonical material. When today we speak of 'computing', our understanding of this word is strongly preshaped by the paradigm of symbolic computing. But brains work differently from digital computers, and so will future non-digital hardware systems, neuromorphic or otherwise. I was using the term 'generalized computing' (GC) as an umbrella term for any approach to engineer physical systems which aims at exploiting a not *a priori* bounded choice of physical phenomena for 'computing'—just as cleverly, opportunistically, and resource-efficiently as biological brain evolution did. In order to establish GC as an engineering science—a faraway goal—new theories have to be developed which formally codify a conceptualization of 'computing' that is as different from the DC paradigm as a brain is from a desktop computer, and maybe even more different. This is a voyage into the unknown. But it is not a voyage blindfolded. A wealth of relevant ideas, experimental designs, and formal theories has already accumulated in a diversity of disciplines and historical lines of thinking. But its richness and diversity is certainly confusing. I attempted to find compass coordinates to help navigating in this intellectual heritage.

An exhaustive exploration of all existing relevant insight is beyond anybody's means. I had to constrain my exploration by a number of decisions which reflect my limited knowledge and personal views:

- From the outset I declare a set of five conditions which I deem necessary for a system to be 'computing', namely that it can become physically instantiated, that it operates in time, that it is open to input or output, that some aspects of it must be interpretable in cognitive terms, and that it must admit a semantic interpretation.
- I restrict myself to investigate formal mathematical theories only, which means I ignored a bounty of philosophical and empirical insight.
- As a coarse scheme for organizing my exploration I adopted a simple-minded division between *physical* computing systems, their environment and the interface boundary between the two on one side, and their respective *formal* models on the other side of what I called the 'modeling mirror' (figure 3).
- I divided formal models of computing systems into two classes. How-models capture the 'mechanics' of computing processes, and what-models give accounts of the computing processes in cognitively interpretable categories which admit semantic mappings to the system's environment.
- I limited my selection of modeling approaches to the ones whose underlying host mathematics is logic, probability theory, or dynamical systems theory (L, P and D, figure 4).

In each L, P, D domain I discussed four themes: mutual transformation/translation interrelationships between formalisms; formal semantics; how time is formalized; and how constructs within a formalism are hierarchically organized. This choice is undoubtedly influenced by my lifelong exposure to DC theory. Except maybe for time, these themes are central coordinates for theorizing in digital computer science. All the non-digital information processing models that I visited reveal that these themes can be elucidated in significantly different ways than we know it from DC. What we see in those other domains can however be called 'computing' too, on the grounds that

- The respective authors call it so;
- Much of this literature relates concretely or by allusion to brains and human cognition, which is the 2370 year old root of today's intuitions about 'computing';
- All of the visited formalisms and models lend themselves to solve practically useful information processing tasks;
- All satisfy the five necessary conditions that I posited (physical realizability, temporality, I/O, cognitive and semantic interpretability).

Theory development for DC is mature and unifying meta-views are canonical, which allowed me to inspect this domain almost in its entirety. In the probabilistic and dynamical systems domains, the untamed diversity of formal models forced me to limit the coverage to quite narrow subdomains, namely sampling-based (SPPD) and ODE based modeling methods, respectively. Here is a summary of what I saw on my journey.

Interrelationships between formalisms. DC how-formalism (in particular, programming languages and abstract models of computing machines) can be transparently sorted into the Chomsky hierarchy, with 'higher' formalisms being able to simulate all 'lower' ones. Within the highest-level class of formalisms (the Turing-equivalent ones), how-formalisms can also be informally ordered according by to how far they abstract away from the 0–1 switching of binary circuits toward cognitive interpretability. What-formalisms are logic formalisms. They can likewise be ordered according to their formal intrinsic semantic expressiveness. These ordering systems tie the many subtheories of DC into 'the' theory of DC which fits into a single textbook. This crystalline transparency sets intimidating standards for attempts to build theories for other sorts of 'computing'.

The how-models of SPPD are sampling algorithms in cases where the SPPD models are physically realized on digital machines; or they are mathematical models of stochastic physical processes when the target hardware is, for instance, analog spiking neuromorphic microchips or DNA computers. The formalisms in which one specifies sampling algorithms or neural or DNA random dynamics stem from different background mathematics, which makes it hard to analyze their mutual relationships; and if one considers sampling independently from the generating mechanisms only in the light of the resulting sample point sequences, there is yet no generally adopted ordering criterion to compare sampling sequences (statistical efficiency might serve as such a criterion). I do not think however that finding useful ordering principles is inherently impossible: an invitation for future research. The what-formalism (singular) for SPPD is the textbook formalism of probability theory, with the cognitively interpretable concept of a probability distribution at its center.

In ODE modeling, the how-formalism is the textbook formalism of ODEs. The how-models are concretely spelled-out systems of ODEs which specify physical computing hardware. Physicists would state that any kind of deterministic physical system can be described by ODEs to arbitrary degrees of approximation, possibly replacing PDEs by finite-element models. Thus the class of ODE computing models is as diverse as one can think of physical computing systems. This is an open-ended diversity, which makes it seem that finding a unifying ordering principle for ODE models is impossible. What-models arise in ODE modeling through the identification of qualitative phenomena like fixed points, attractors and bifurcations, to name the ones which today are being most widely used as anchors for cognitive interpretations. ODEs are a well for an unbounded number of other qualitative phenomena that await their discovery. This may imply that a general what-theory is impossible for ODE modeling.

It becomes clear that in the DC, SPPD and ODE domains alike we face voluminous assortments of formalisms and models. Finding criteria and mathematical methods to relate them to each other is a sign of a domain's integrity and maturity. Only the millenia-old field of symbolic computing can claim maturity in this regard. Here is the upshot of this high-altitude flight over the theory landscapes in DC, SPPD, and ODE:

- DC offers a wealth both of how- and what-formalisms, which are transparently and comprehensively interrelated.
- SPPD has a single 'master' what-formalism and a yet unsorted multiplicity of how-formalisms and models.
- Conversely, ODE modeling has a single how-formalism but what-formalizing may be principally open-ended.

Formal semantics. Formal semantic relationships are mathematically defined relations between syntactic *expressions* (formulae, theorems, …) on the one end and their *referents* (interpretations, denotata, models, …) on the other end. The objects on both sides (expressions and referents) must be mathematically defined. In the semantic theory of formal logic, the expressions are formal symbol sequences with rigorously defined grammars, and the referents are certain set-theoretic structures. Furthermore, a semantic theory usually specifies how syntactic constructions which build up increasingly complex expressions are reflected in compound-building operations in the referent domain. The word 'semantics' is standardly reserved for logic formalisms. In this article I extend its use to discuss relationships between two formal domains that are defined in probability and dynamical systems theory.

Specifically, I consider 'semantic' relationships between cognitively interpretable constructs in what-formalisms (on the expressions side) and formal models of objects, facts, situations processes etc in mathematical models of a computing system's environment (on the referents side). Such semantic relations create bridges between segments **a** and **c** in figure 3. I find it a mandatory component of any theory of 'computing' that it allows one to express how what happens 'cognitively' inside a computing system relates to conditions in its environment: this is the key to make a computing system understandable and practically employable by human users.

Every logic that serves as a DC what-formalism comes equipped with a formal semantic. It precisely defines how the formulas that can be written down in the logic (segment **a**) become *interpreted* in formal models (in **c**) of environments. These formal models are cast as certain richly structured sets called S-structures. Formal logics offer no mathematical account of the interface boundary **b** between a computing system and its environment. In the view of logicians, the symbols and expressions written down in a logic formalism are directly *denoting* something in the S-structures, without an intermediary exchange of 'data' or 'signals'. When physical digital computers (segment α) are used in the physical world γ, it is up to the physical (human or automated) user of the computer to establish a real-world semantic linkage between the computer and its task environment by providing appropriate input and interpreting the computer's output appropriately. Formal logic is blind to the interface boundary and does not capture how the semantic relationship is constructed, recognized or verified.

In contrast, probability theory (the singular what-formalism in all probabilistic modeling, not only SPPD) captures 'observations', 'data' or 'signals' in its core constructs, namely random variables and sample spaces. Random variables are the formal correlate of measurement or observation procedures and apparatuses. Sample spaces are sets made from the data values that these observation procedures may deliver; they thus perfectly coincide with segment **b** in figure 3. Like in logics, the formal models of the environment are richly structured sets which additionally are endowed with a probability measure, called probability spaces. Probability spaces constitute segment **c**. Random variables can be regarded as semantic operators in that they connect segments **a** with **c** through **b**. This connection is unidirectional. While random variables capture the *input* given by the environment to inform the shaping of probability distributions which are modeled in **a**, there is no provision for capturing any impact which an *output* from a probabilistic 'reasoning' process modeled by the how-formalism in **a** could have on the probability space.

In ODE modeling the semantic situation is again different. In dynamical systems modeling the computing system and its environment are seen as coupled subsystems within a single dynamical agent-environment system. Both subsystems **a** and **c** are described through ODEs. They are bidirectionally coupled through shared variables which constitute the interface boundary **b**. The mathematical substrate for how-modeling both the computing 'agent' as well as its environment are vector fields—there is no difference in mathematical kind between the modeled inner and outer worlds. In dynamical systems oriented research where agent-environment interactions are studied, these interactions are not understood as 'semantic'. In the epistemological view of constructivism and enactivism, and according to some authors in the situated agents tradition (behavior-based robotics, 'new AI'), these models are even expressly advanced as anti-semantic, denying that it makes sense to consder mental 'representations' of outward givens.—However, if one adopts the view which I take in this article, namely to locate the formal intrinsic semantic relationship between what-models and models of the environment, the semantic question arises again. Concretely, one would ask how cognitively interpretable dynamical phenomena (attractors, bifurcations, ... in segment **a**) can be formally connected to environment models. This is a deeply ploughed field but the ploughing tracks crisscross wildly since a theory compass is missing.

These findings reveal that there are quite different ways to think about semantics. 'Semantics' is not a universally defined and agreed concept and every philosopher will understand it in a different way. This opens many degrees of freedom for developing GC theories. Here is the upshot of my high-speed drive through the semantics challenge in DC, SPPD, and ODE:

- In DC and SPPD, accounts of intrinsic semantics are firmly established in fundamental mathematical definitions. They specify how cognitively interpretable formal constructs in models of 'computing' system relate to formal models of their environments. Nothing comparable is yet available for ODE modeling.
- In DC and SPPD, the mathematical 'substrate' of environment models is different in kind from the substrate of the cognitively interpretable constructs in agent models, namely set-theoretic structures versus

symbolic configurations and probability distributions, respectively. In ODE modeling, both sides are made of the same mathematical material, namely vector fields.

- SPPD and ODE modeling offers canonical constructs for the interface boundary (data, signals) between a computing system and its environment; logic does not.

Formal time. Regardless how one understands the nature of 'computing', one thing seems inevitable: physical computing systems need time to 'compute'. Any formal theory of any kind of computing should also model the computing system's time, certainly in its how-formalisms and models. They are the springboards from which system engineers jump across the modeling mirror and build machines, guided by formal how-models in the back of their minds. In the three domains that I inspected, time is modeled in interestingly different ways in how-models. Even more interesting are the differences between DC, SPPD and ODE with regard of how the physical time that is modeled in how-models relates to the 'conceptual' time that is (or is not) formalized in what-models.

In DC how-formalisms, time enters in the form of update steps where one symbolic configuration is transformed into the next. These steps unfold into nonzero-duration increments of physical time on the physical digital machines. The physical duration can be longer or shorter (depending on the clock speed and the degree of parallelization) and it is not modeled in typical how-formalisms (real-time operating systems excepted). When one sees these update steps in the light of logic-based what-formalisms—the light shining from Aristotle and Turing—they are not seen as temporal at all, but as inferential. The next symbolic configuration follows logically from the previous. The verb 'follow' is fascinatingly ambiguous, with one of its meanings being temporal succession and another one being logical implication. There are more words which have a temporal and a logical-inferential side, for instance 'consequence', 'conclude'. Seen from this angle, the history of DC could be summarized like this: first it took philosophers and mathematicians almost 2300 years to strip logical inference off from the physical brain's physical time, a process which became finalized in Tarski's reconstruction of logical implication in terms of static inclusion relations between classes of S-structures. Then temporal succession was re-introduced by Turing in the form of an ordered sequence of discrete symbolic configuration update 'steps'. Ultimately, electrical engineers and chip manufacturers rejoin physical time by realizing these steps within the clock cycles of digital microchips.

In SPPD how-models time can be cast as a discrete sequence of update steps (in sampling algorithms destined for execution on digital machines), or it can be cast as the continuous time-line $t \in \mathbb{R}$ when the sampling process is modeled for use in non-digital hardware, in particular in analog spiking neuromorphic microchips. In both variants there appears a fundamental difference to DC thinking. The cognitively interpretable constructs (the probability distributions handled in what-models) need nonzero timespans to be grown. The longer a sampling process carries on, the more precisely it defines its distribution. The cognitively interpretable constructs are smeared out over time. If one sees a 'computing' process as reflecting some aspect of a succession of 'mental states' (which Turing did—and we all stand on his shoulders), then these mental 'states' become defined only across time, with their constituting components (the representations of distributions) growing, overlapping in time, decaying. This is in stark contrast to the DC view where at each time point the 'mental state' is perfectly and completely defined by a symbolic configuration, and future configurations do not gradually grow out of previous ones but are created in their completion immediately and discontinuously.

ODE how-formalisms (ODE specifications of biological or engineered computing systems) use the real line $\mathbb{R}$ as their model of time. The temporal evolution $\dot{\mathbf{x}}$ of the state vector $\mathbf{x}$ progresses smoothly and with perfect real-valued precision. This makes it *possible* for engineers and neuroscientists to directly check the adequacy of their ODE how-models in matching their physical target systems by measuring physical system variables with appropriate measurement apparatuses. It also makes it *necessary* for engineers to bidirectionally match the physical timescales in their analog hardware with the formal timescales in their design models. In the light of own experience with analog neuromorphic hardware, I consider it a decisive challenge for future GC engineering to master timescale spreads and timescale interactions both in theory and in physical devices and systems. The problem of setting up (hierarchies of) appropriate *time constants* is intimately connected to the theme of hierarchical organization of 'computing' and will be discussed below.

Most of the cognitively interpretable constructs treated in what-formalisms and models (attractors, bifurcations, etc) are defined by topological-geometrical phenomena in phase portraits. 'Speed' becomes factored out in these analyses: phase portraits are made from trajectory lines, and the information how 'fast' the state evolution progresses along these lines is discarded. This is a parallel with DC and probabilistic modeling, whose cognitively interpretable what-constructs (symbolic configurations, distributions) are likewise atemporal.

Human cognition proceeds in time, physical computing systems run in time, and we all share a primal understanding of 'time'. One thus would expect a universal, intuitively graspable capture of time in 'computing' theories and formalisms. But we find diversity and detachment from intuition. Here is the upshot of my meandering trip through the time modeling challenge in DC, SPPD, and ODE:

- Time in how-formalism is cast as a sequence of discrete update steps (in DC and in sampling algorithms) or as continuous (in dynamical systems models of neural sampling processes and ODEs).
- Matching formal time in how-formalism with physical time in the modeled computing systems is arbitrary in DC, and well-defined and measurable in ODE. In SPPD both occurs depending on whether the sampling becomes physically instantiated on digital or non-digital physical systems.
- The cognitively interpretable constructs that are commonly expressed in the what-models of DC, SPPD and ODE are almost all a-temporal, which is somewhat bewildering since human cognition is temporal.

Hierarchical structuring of formal constructs. A characteristic of human cognitive processing is *compositionality*: we can compound syllables into words into phrases, bind noses, eyes and mouths into faces, plan complex plans that unfold in cascades of sub-plans to reach sub-goals—we can think *complex* thoughts. This ability has been claimed constitutional for human intelligence, and it seems natural to request the same from any full-scale 'computing' system and its theoretical models.

In DC how-formalisms, hierarchies appear in two main ways. First, the symbol structures which are step-wise constructed when how-models are 'executed' are organized as hierarchically nested composites. Second, this *syntactic* compositionality of symbolic configurations is typically tied in with a *procedural* hierarchical organization of the runs of programs or formal machine models: symbolic substructures are built in nested subroutines, with the effect that substructures correspond to sub-intervals in processing time. Writing a non-trivial computer program amounts to breaking down the global input-to-output functionality imposed by the given task into a nested sequence of intermediate goals and subgoals. Much sweat is spent by students in software engineering classes when they learn to write neatly structured code.

In DC what-formalisms (that is, logic formalisms) the symbolic expressions typically contain nested functional expressions at their compositional root level, which can be composed into arbitrarily complex formal structures with numerous kinds of logical operators, depending on the specific logic. The functional expressions and the logical operators are cognitively interpretable as object concepts and fact assertions, respectively.

In the SPPD domain, how-models describe sampling procedures. They are not typically hierarchically structured.

The SPPD what-formalism is the single canonical formalism of probability theory. The primary cognitively interpretable construct are distributions. Distributions can be sees as compositional in several ways. First, a number of popular stochastic spiking neural network architectures are layered, in analogy to the peripheral-to-central processing organization found in human brains. When used in (for example) in face recognition tasks, samples collected from low-level neurons are considered to represent local visual features (like colors, edges or dots) of input images, whose information becomes increasingly combined and globalized in higher layers (from edges to contour segments to eyes to faces). The hierarchization here is defined through the processing architecture rather than being an intrinsic characteristic of the participating distributions. Second, in Bayesian models of cognitive information processing, 'higher' distributions arise as hyperdistributions, that is, distributions of distributions. Third, a fundamental textbook operation on distributions is to combine them into product distributions (where the component distributions remain statistically independent) or joint distributions (where the component distributions interact and become statistically dependent on each other). Conversely, high-dimensional distributions can sometimes be more or less precisely *factorized* into low-dimensional component distributions.

In ODE systems, a (I would say *the*) key to hierarchical structuring of qualitative phenomena are timescale hierarchies. While elementary analyses of the interaction between slow and fast system variables are important for the how-modeling needed for hardware or brain modeling, the full power and complexity of timescale hierarchies would become manifest only when they were combined with geometrically defined, cognitively interpretable qualitative phenomena like attractors etc. According to a widespread view in the cognitive and neurosciences, the hierarchy of timescales is a mirror of the compositional hierarchy of cognitive 'representations'. This leads to the pervasive idea that cognitive architectures are layered structures in which 'higher' processing layers evolve more slowly than 'lower' layers. Working out this perspective in conjunction with other qualitative phenomena however remains a topic for future research.

The essence of compositional hierarchies in L, P, D is that elements that are higher in the hierarchy are made from elements from lower layers. This 'made from' relation, however, can mean quite different things in different approaches to modeling 'computing'. Here is the upshot of my dash into the thickets of hierarchical structuring phenomena of DC, SPPD, and ODE:

- The symbolic configurations in DC how- and what-formalisms are structured in a static-syntactic Lego-brick kind of hierarchical compositionality.

- Probability distributions can be seen as hierarchically structured in several ways, all of which are not merely syntactically defined but can be more appropriately understood by observing that component distributions shed some of their statistical information into the compound distribution.
- A key to hierarchic organization in complex dynamical systems is a hierarchy of timescales, which induce a hierarchically nested sequence of processing modes.

Memory: an knot of time, space, information, compositionality, semantics. The theory coordinates discussed in the previous paragraphs are not independent from each other. One can feel their entanglement when one starts thinking about *memory*.

The word ‘memory’ has two semantic anchors which to a certain extent can be kept apart.

The first is ‘memory’ as a place where information is stored and/or as mechanisms to encode, write, address, retrieve and decode. Such *information storage* aspects shine up in many places in the wider ‘computing’ fields, for instance in the tape in a Turing machine, the addressable memory registers in a von Neumann machine, regions in the human brain, neurophysiological models of biological working or long-term memory, etc. This information storage aspect is in turn connected to the question of how ‘information’ should be defined in the first place and then what it means that it becomes en-, re-, de-coded. And this is furthermore leading to questions of what ‘learning’ is, if learning is seen as acquiring, encoding and storing information. I have deliberately left out an investigation of information and learning in this article, but only for the bad reason of not making it even longer.

The information storage aspect of memory receives much attention in the discussion around the von Neumann bottleneck and the hopes placed on in-memory computing. While of obvious technological importance, I do not think that this specific discussion is central for developing a GC theory. At least, it seems not central for the grand role model of theorizing about computing, DC theory. The von Neumann bottleneck is an annoying side-effect of just one among dozens equivalent abstract models of computing, namely of the random access machine (the abstract mathematical version of von Neumann architectures). The Turing machine is even more crippled than random access machines because memory tape cells are not even addressable but must be crawled to; some other models of computing could be said to reflect in-memory computing (surely cellular automata, but term replacement systems could also be viewed as in-memory models), and yet other more abstract general models of computing (lambda calculus, recursive functions) do not have a notion of memory-as-storage-place at all.

The second semantic dimension is ‘memory’ as *information transfer through time*. A physical computing machine is a physical system, and it is an unquestioned presupposition of classical physics that one can assign a *state* $\mathbf{x}(t)$ to any physical system at any time t. The state $\mathbf{x}(t)$ at time t contains all the traces of the past that the system will ever be able to use in the future. In a purely mathematical sense, $\mathbf{x}(t)$ simply *is* the memory of the system at time t. However, when one analyzes a system in the light of information processing, a multitude of questions jump up which are as difficult to be stated precisely as they are difficult to answer. In naive wording, core questions are: what is the ‘content’ of memory? How is its content encoded? When was it encoded? By what mechanism? How can memory items be addressed? Retrieved? Decoded? How do memory traces shape the ongoing evolution of the system?—When memory traces are recognized as intrinsic components of system states, and system states evolve over time, the next-level question is: how do memory traces evolve over time? How are they fading or prevented from fading? How is the creation of memory contents related to learning? How are hierarchies of memory structures and processes linked to timescale hierarchies in the dynamics of the information processing system? Such questions have given rise to streams of researches in mathematics and physics, the cognitive and neurosciences, as well as in machine learning and AI. Here is not space enough to unfold them. In a preliminary and partial survey we have made an initial attempt to reconnoiter such and other aspects of time- and memory-scales in information-processing systems [Jaeger *et al* 2021]. Here is a summary of insights obtained from that survey and from material collected in section 2.3:

- Some information processing tasks (often in signal processing and in reservoir computing) only need memory which fades out on a single timescale. Many tasks of higher cognitive complexity (for instance in speech and language processing or action planning) however need to encode incoming information into hierarchical memory structures where higher-level memory items often semantically represent longer-timescale external phenomena.
- System states $\mathbf{x}(t)$ and system episodes $(\mathbf{x}(t))_{T\leqslant t\leqslant T'}$ do not per se represent or encode memory contents. It is up to external interpreters or system-internal decoding mechanisms which specific mathematical features of states or trajectories become attributed as carrying relevant information. Here ‘information’ is not used in the well-understood but purely probabilistic sense of Shannon information but requires other, less well-developed theoretical frameworks which add a semantic dimension to this concept.
- For computing systems that utilize memory beyond a single transient timescale it seems useful and maybe necessary to formally cast them as *spatio-temporal* systems. System models then can assign hierarchically

structured ‘places’ to likewise structured memorized contents. These ‘places’ can be defined in many ways, be it metrically or in graph topologies or otherwise. They can become explicit in the model, like in the Turing tape, cellular neural networks or brain areas where dedicated formula elements denote ‘places’. They can also be implicit, like in lambda calculus where the syntactic structure of a model formula itself coincides with the topological (here: tree) structure of the modeled system.

The title of this section is *staking out the ‘computing’ theory landscape*. I could cover the dominion of DC almost in its entire extension, though of course with simplifications and omissions. This was possible because theoretical (digital) computer science is mature, unified and canonized; thus all I had to do is to map *the* DC theory to the organigram of figure 3. For probabilistic and dynamical systems oriented models of ‘computing’, unified meta-views are not in sight. In order to not get lost I selected small sectors of them, namely sampling-based computational methods to represent probability distributions, and ODE modeling. But even within this limited angle of vision, landmarks and signposts came into view which invite us to explore ‘computing’ in many more directions than those of the digital-symbolic paradigm. Here is my personal grand total of this first expedition:

- The way of how one *can* conceptualize ‘computing’ is decisively preshaped by the choice of mathematical host formalism (here: logic, probability theory, dynamical systems).
- Different conceptualizations of ‘computing’ grow around different aspects of human cognition (logical inference, probabilistic reasoning and degrees of belief, continuous sensor–motor coordination).
- DC will forever remain the emperor over the entire ‘computing’ realm in the sense that digital computer programs can carry out logic inference; logic (together with set theory) can express all of mathematics; all formal how-models of ‘computing’ are mathematical; hence digital computers can simulate all other formal procedural specifications of ‘computing’. However, this emulation can become prohibitively inefficient with regards to runtimes, energy consumption and microchip complexity.
- Procedural formalisms (I called them how-formalisms) are the springboards for engineers from which they jump across the modeling mirror, building physical computing systems which realize the formal specifications. Depending on the chosen mathematical substrate, different limitations and opportunities for physical designs arise. Digital system engineers *must* build hardware based on finite-state switching operations and memory mechanisms to stably store switching states for very long times. Once they know how to build such machines, they *can* capitalize on the full powers of symbolic computing theory. System engineers informed by sampling how-formalisms *must* find ways to harness physical stochasticity. Once they master this task, they *can* build machines which realize already existing, general models of probabilistic inference (graphical models, Bayesian networks, Boltzmann machines, simulated annealing optimizers). At present, physical randomness has been made exploitable for sampling only in limited ways in DNA computing (note that quantum computing exploits randomness not by sampling and was not covered in this article). System engineers guided by ODE models of cognitive processing *should* learn to realize an ever growing repertoire of ODE models in physical dynamics. Then they *could* build machines which re-play the cognitive mechanisms that have been discovered and will be discovered in the wider cognitive and neurosciences, as qualitative phenomena in dynamical systems. They could achieve even more if a dynamical systems understanding of ‘computing’ disengages itself from its current focus on ‘brain-inspired’ neuromorphics.
- How-theories of ‘computing’ include, implicitly or explicitly, a formal model of *time*, which in turn co-determines which cognitive operations can be captured in corresponding what-formalisms, and how. The importance of analyzing how ‘computing’ processes are structured in time is, in my opinion, largely under-appreciated. The same could likely be said about *space*, a theme which I decided to leave out in this article. Temporal and spatial phenomena are closely coupled in physical systems. This demands a discussion which I postpone to another occasion.

3. Conclusion: where one might start looking for a generalized theory of ‘computing’

Despite the limited scope of reconnaissance in the previous section I hope that what I spotted gives some helpful orientation in the search for a generalized theory of ‘computing’. In this concluding section I tentatively suggest some ideas toward this end.

The mission for GC would be to transform the not-so-new but still vague idea of exploiting ‘physics directly’ into an engineering discipline with a dependable formal theory foundation. This will require a collaboration between contributors who today rarely work together—materials scientists, theoretical physicists, microchip engineers, neuroscientists, cognitive scientists, AI and machine learning experts, computer scientists, mathematicians and epistemologists. They can only start talking with each other after they have agreed on a shared language, and they can only precisely understand what they are talking about after having developed a mathematical foundation underneath this language.

In my opinion, which I tried to substantiate in this serpentine article, any theory about any kind of physical systems can qualify as a theory of 'computing' only if core constructs in that theory can be linked to human cognitive processing.

In my review of the D, P and L domains I placed the formal correspondents of cognitive entities into the center of what I called what-formalisms (in segment **a** in figure 3), and inspected the formal semantic links which connect those formal constructs with formal models of environments (segment **c**). Across the modeling mirror, what is formally represented in models of computing systems and models of environments should have reflections in the corresponding physical computing system (in α) and its physical environment (in γ). Furthermore, the physical computing system should enjoy physical semantic relations with its environment which make the diagram

$$\begin{tikzcd}
\textbf{(a)} \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & \textbf{(c)} \arrow[d, leftrightarrow] \\
(\alpha) \arrow[r, leftrightarrow] & (\gamma)
\end{tikzcd}$$

commute. This implies that there should be a correspondence between the cognitively interpretable constructs in what-formalisms (proceduralized in how-formalisms) on the one hand, and phenomena in physical computing systems on the other hand. This is in fact what we could observe in DC (discrete symbol operations corresponding with binary switching dynamics), in sampling-based probabilistic models (formal sample points corresponding to, for example, spikes in neural substrates or DNA sniplets in DNA computing), and in ODE modeling (for instance, periodic attractors realized in oscillatory electronic circuits).

Thus, when we want to capitalize on *any* kind of computing-enabling physical phenomenon in GC machines, we have to address the question, *what is it in human cognition that allows us to mentally represent 'any' kind of physical phenomenon?* I am afraid that human intelligence cannot grasp 'any' kind of physical phenomenon. This version of the GC mission statement seems too strong. In the light of findings in the previous section, we should more narrowly ask: *what temporal physical phenomena, which can be coupled into increasingly compounded complex phenomena, can be cognitively grasped?* or its mirror twin: *what cognitive phenomena make us perceive and think about physical processes as being constituted of coupled sub-processes?*

This very question was answered by Alan Turing in a specific way in his celebrated article [Turing 1936] which can be regarded as the founding document of DC. In this article he developed the concept of what was later called the Turing machine—the quintessential model of a digital computer. Turing was a mathematician and in this article he wanted to settle a foundational question of formal logic. But instead of presenting a purely mathematical abstract theory, he pondered on how a *human* 'computer' carries out mathematical calculations. He was indeed using that word, 'computer', to denote a human operator, and he referred to a 'computer' by the personal pronoun 'he'. In order to map human cognitive processing to operations of mathematical calculation Turing committed to a decisive abstraction. He decreed that 'states of mind' should be viewed as discrete and identifiable. This led him to model cognitive processes through finite sets of symbols which can be compounded in composite structures—the rest is DC history.

Following in Turing's footsteps, but aiming for greater generality, I think it is a good starting point to contemplate a richer, more dynamical picture of 'states of mind'. I would think that such a generalized picture would view 'states of mind' not as static, discrete entities but as dynamically changing, continually in flux and morphable from one into another.

I want to illustrate this view with a concrete example. Attach LEDs to some joints and extremity ends of a human volunteer. Let this person move in an unlit room such that only the light traces of these LEDs are visible. A human observer can then readily recognize whether the volunteer walks, runs, jumps, waltzes, or engages in any of hundreds of human motion patterns. The observer can also decompose the overall dynamic pattern presented by all the LEDs into subpatterns, for instance focusing on the right arm's motion.—This is an example from visual sensory processing. But the scope of our twin question is far wider and could be filled with examples from other sensory modalities; not only from perception but also from action generation; or from the entire mental experience of being present in a dynamic environment; or even from a mathematician sitting still in deep thought in front of a white sheet of paper—mathematical thinking arguably being just another way of re-experiencing dynamical situatedness [Lakoff and Nunez 2000].

When queried *after* the observation, the human observer will be able to give an account of what s/he observed in discrete words: the LED pattern was a case of 'walking', 'running', etc. But *during* the observation, *in the process of experiencing* the visual input, the human observer's mind will not jump discretely between a finite set of mental states. Instead, his/her brain's processing will be entrained by the continual visual stream and his/her mental state will evolve continually.

Cognitive scientists have long since been investigating complex, gradually morphable mental representations of dynamical phenomena, developing comprehensive theory frameworks like *fluid concepts* [Hofstadter 1995] or *radial categories* [Lakoff 1987], or exploring complex motion pattern representations [Bläsing *et al* 2009, Tervo *et al* 2016]. These are random pointers only; there are far more investigations of this kind in the wider cognitive sciences.

Theories in cognitive science, in particular in its experimental branches, are however often articulated only in natural English. If one searches for mathematically worked-out formalisms and theories that might be useful for registering such dynamical mental states, one can find them outside the core cognitive sciences. I personally have felt instructed by mathematical models in (human and animal) *motion science* where one objective is to formally describe complex motor patterns and analyze how they can be perceived and controlled [Hogan and Flash 1987, Thoroughman and Shadmehr 2000, d'Avella *et al* 2003]. My LED-tracing example was borrowed from recent work in this line [Land *et al* 2013]. Grenander's *pattern theory*, especially in the transparent rendering of David Mumford [Mumford 1994, 2002], offers a thoroughly formal account of how (primarily spatial/visual) 'patterns' which are emerging in complex physical systems can be generated, compounded, transformed and encoded. A classical subfield of AI, *qualitative physics* [Forbus 1988] (closely related: *naive physics*, *qualitative reasoning*) explores logic-based formalisms which capture the everyday reasoning of humans about their mesoscale physical environment. Insights gained in the fields of *emergent computation* [Forrest 1990] steer attention to the powers of collective phenomena in dissipative systems, where macrolevel phenomena 'self-organize' from the interactions of microlevel components. Machine learning and data mining methods for detecting *concept drift* [Gama *et al* 2013] offer statistical characterizations of how data streams change qualitatively over time, including recent methods which exploit hierarchical structuring of distributions [Hammoodi *et al* 2018]. Finally, recent propositions to develop a theory of *stream automata* [Endrullis *et al* 2019] aim at extending the classical theory of finite-state automata to infinite data stream processing. These and other sources of mathematical inspiration remain to be surveyed and connected.

At this moment I do not see a single compelling ansatz which holds promise to become worked out into a formal theory that could answer the twin question posed above. In a separate manuscript (in preparation) I describe in more detail why I think that such a system of interconnected formal theories can be rooted in the concept of *dynamical modes*, which generalize the bi-stability modes of digital switching transistors.

I believe that we are facing a quite fundamental challenge, and that we are painfully missing an adequate mathematical language. Newton and Leibniz devised calculus to capture *continuous motion*. Kolmogorov and his predecessors developed probability theory to capture the *information conveyed by empirical observations*. Tarski and his predecessors cast formal logic in its modern shape in order to capture *rationally derivable truth*. If I were pressed to condense that twin question of GC into a similarly momentous three-word phrase I would say that we have to capture *gradual qualitative change*. I believe that this asks us to discover a profoundly new mathematical language, a new branch in the tree of mathematics which grows between probability (for 'gradual'), logic (for 'qualitative') and dynamical systems (for 'change'). I sometimes tell my students that I hope to live to the day when one among them finds it.

Acknowledgments

I am indebted to a very constructive anonymous reviewer who helped me to see and fill a number of glaring gaps in the first version of this paper. I gratefully acknowledge the financial support from the CogniGron research center and the Ubbo Emmius Funds (Univ. of Groningen), and from the EU H2020 project MeM-Scales (Grant Agreement 871371).

Data availability statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ORCID iDs

Herbert Jaeger https://orcid.org/0000-0002-1230-4963

References

Abraham R H and Shaw C D 1992 *Dynamics: The Geometry of Behavior* (Reading, MA: Addison-Wesley Developers Press)

Ackley D H, Hinton G E and Sejnowski T J 1985 A learning algorithm for Boltzmann machines* *Cogn. Sci.* **9** 147–69
ACM 2018 Turing award laudatio for deep learning pioneers https://amturing.acm.org/award_winners/bengio_3406375.cfm (accessed June 2020)
Adamatzky A (ed) *Advances in Unconventional Computing Vol. 1: Theory* 2017a (Basel: Springer)
Adamatzky A (ed) *Advances in Unconventional Computing Vol. 2: Prototypes, Models and Algorithms* 2017b (Basel: Springer)
Adamatzky A 2018 Towards fungal computer *Interface Focus* **8** 20180029
Adamatzky A, De Lacy Costello B and Asai T 2005 *Reaction Diffusion Computers* (Washington: Elsevier)
Albus J S 1993 A reference model architecture for intelligent systems design *An Introduction to Intelligent and Autonomous Control* ed P J Antsaklis and K M Passino (Dordrecht: Kluwer) ch 2 pp 27–56
Amit D J, Gutfreund H and Sompolinsky H 1985 Spin-glass models of neural networks *Phys. Rev.* A **32** 1007–18
Andrae A and Edler T 2015 On global electricity usage of communication technology: trends to 2030 *Challenges* **6** 117–57
Ashby W R 1952 *Design for a Brain* (New York: Wiley)
Babloyantz A and Lourenço C 1994 Computation with chaos: a paradigm for cortical activity *Proc. Natl Acad. Sci.* **91** 9027–31
Baddeley A 2003 Working memory: looking back and looking forward *Nat. Rev. Neurosci.* **4** 829–39
Basye K, Dean T and Kaelbling L P 1995 Learning dynamics: system identification for perceptually challenged agents *Artif. Intell.* **72** 139–71
Beer R 1995 Computational and dynamical languages for autonomous agents *Mind as Motion: Explorations in the Dynamics of Cognition* ed R Port and T van Gelder (Cambridge, MA: MIT Press) ch 5 pp 121–48
Berner C *et al* 2019 Dota 2 with large scale deep reinforcement learning (arXiv:1912.06680)
Besold T *et al* 2017 Neural-symbolic learning and reasoning: a survey and interpretation (arXiv:1711.03902)
Bizzi E and Mussa-Ivaldi F A 1995 Toward a neurobiology of coordinate transformations *The Cognitive Neurosciences* ed M S Gazzaniga (Cambridge, MA: MIT Press) ch 31 pp 495–506
Bläsing B, Tenenbaum G and Schack T 2009 The cognitive structure of movements in classical dance *Psychol. Sport Exerc.* **10** 350–60
Boahen K 2017 A neuromorph's prospectus *Comput. Sci. Eng.* **19** 14–28
Bose S K, Lawrence C P, Liu Z, Makarenko K S, van Damme R M J, Broersma H J and van der Wiel W G 2015 Evolution of a designless nanoparticle network into reconfigurable Boolean logic *Nat. Nanotech.* **10** 1048–52
Bournez O and Pouly A 2018 A survey on analog models of computation (arXiv:1805.05729)
Brooks R A 1989 The whole iguana *Robotics Science* ed M Brady (Cambridge, MA: MIT Press) pp 432–56
Brooks R 1991 Intelligence without representation *Artif. Intell.* **47** 139–59
Buesing L, Bill J, Nessler B and Maass W 2011 Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons *PLoS Comput. Biol.* **7** e1002211
Buongiorno F 2019 From the extended mind to the digitally extended self: a phenomenological critique *Aisthesis* **12** 61–8
Burgin M and Dodig-Crnkovic G 2013 From the closed classical algorithmic universe to an open world of algorithmic constellations *Computing Nature: Turing Centenary Perspective* (Berlin: Springer) pp 241–53
Buzsáki G and Chrobak J J 1995 Temporal structure in spatially organized neuronal ensembles: a role for interneuronal networks *Curr. Opin. Neurobiol.* **5** 504–10
Chen T *et al* 2020 Classification with a disordered dopant-atom network in silicon *Nature* **577** 341–5
Chen T, Bobbert P A and Wiel W G 2021 1/*f* noise and machine intelligence in a nonlinear dopant atom network *Small Sci.* **1** 2000014
Chuang A T, Margo C E and Greenberg P B 2014 Retinal implants: a systematic review: table 1 *Br. J. Ophthalmol.* **98** 852–6
Clark A 2013 Whatever next? Predictive brains, situated agents, and the future of cognitive science *Behav. Brain Sci.* **36** 181
Coecke B 2012 The logic of quantum mechanics-take II (arXiv:1204.3458)
Coulombe J C, York M C A and Sylvestre J 2017 Computing with networks of nonlinear mechanical oscillators *PLoS One* **12** e0178663
Csaba G, Papp Á and Porod W 2017 Perspectives of using spin waves for computing and signal processing *Phys. Lett.* A **381** 1471–6
d'Avella A, Saltiel P and Bizzi E 2003 Combinations of muscle synergies in the construction of a natural motor behavior *Nat. Neurosci.* **6** 300–8
Davies M *et al* 2018 Loihi: a neuromorphic manycore processor with on-chip learning *IEEE Micro* **38** 82–99
de Castro L N 2006 Fundamentals of natural computing: an overview *Phys. Life Rev.* **4** 1–36
Dellaert F, Fox D, Burgard W and Thrun S 1999 Monte Carlo localization for mobile robots *Proc. IEEE Int. Conf. on Robotics and Automation*
Deneve S 2008 Bayesian spiking neurons: I. Inference *Neural Comput.* **20** 91–117
Durstewitz D, Seamans J K and Sejnowski T J 2000 Neurocomputational models of working memory *Nat. Neurosci.* **3** 1184–91
Ebert C 2018 50 years of software engineering: progress and perils *IEEE Softw.* **35** 94–101
Eliasmith C, Stewart T C, Choo X, Bekolay T, DeWolf T, Tang Y and Rasmussen D 2012 A large-scale model of the functioning brain *Science* **338** 1202–5
Endrullis J, Klop J W and Bakhshi R 2019 Transducer degrees: atoms, infima and suprema *Acta Inform.* **57** 727–58
Euler H C R, Boon M N, Wildeboer J T, van de Ven B, Chen T, Broersma H, Bobbert P A and van der Wiel W G 2020 A deep-learning approach to realizing functionality in nanoelectronic devices *Nat. Nanotechnol.* **15** 992
European Commission Author Collective 2009 *Unconventional Formalisms for Computation: Expert Consultation Workshop. Workshop Report* European Commission https://cordis.europa.eu/pub/fp7/ict/docs/fet-proactive/shapefetip-wp2011-12-05_en.pdf
Everhardt A S, Damerio S, Zorn J A, Zhou S, Domingo N, Catalan G, Salje E K, Chen L Q and Noheda B 2019 Periodicity-doubling cascades: direct observation in ferroelastic materials *Phys. Rev. Lett.* **123** 087603
Farhang-Boroujeny B 1998 *Adaptive Filters: Theory and Applications* (New York: Wiley)
Felleman D J and Van Essen D C 1991 Distributed hierarchical processing in the primate cerebral cortex *Cerebr. Cortex* **1** 1–47
Feynman R P 1982 Simulating physics with computers *Int. J. Theor. Phys.* **21** 467–88
Fodor J A and Pylyshyn Z W 1988 Connectionism and cognitive architecture: a critical analysis *Cognition* **28** 3–71
Forbus K D 1988 Qualitative physics: past, present and future *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence* (San Mateo, CA: Morgan Kaufmann Publishers) pp 239–96
Forrest S 1990 Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing networks *Physica* D **42** 1–11
Fredkin E 2013 Discrete theoretical processes (DTP) *A Computable Universe: Understanding and Exploring Nature as Computation* ed H Zenil (Singapore: World Scientific) ch 19 pp 365–80
Freiberger M, Katumba A, Bienstman P and Dambre J 2017 On-chip passive photonic reservoir computing with integrated optical readout *Proc. IEEE Int. Conf. on Rebooting Computing (ICRC2017)* pp 255–8

Frémaux N and Gerstner W 2016 Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules *Front. Neural Circuits* **9** 85

Friston K J, Daunizeau J, Kilner J and Kiebel S J 2010 Action and behavior: a free-energy formulation *Biol. Cybern.* **102** 227–60

Friston K 2005 A theory of cortical responses *Phil. Trans. R. Soc.* B **360** 815–36

Fusi S and Wang X-J 2016 Short-term, long-term, and working memory *From Neuron to Cognition via Computational Neuroscience* ed M Arbib and J Bonaiuto (Cambridge, MA: MIT Press) pp 319–44

Gallego G *et al* 2020 Event-based vision: a survey (arXiv:1904.08405)

Gama J, Indrė Ž, Bifet A, Pechenizkiy M and Bouchachia A 2013 A survey on concept drift adaptation *ACM Comput. Surv.* **46** 44

Gerstner W, Kreiter A K, Markram H and Herz A V M 1997 Neural codes: firing rates and beyond *Proc. Natl Acad. Sci.* **94** 12740–1

Goldman M, Compte A and Wang X-J 2008 Neural integrators: recurrent mechanisms and models *New Encyclopedia of Neuroscience* ed L Squire, T Albright, F Bloom, F Gage and N Spitzer (Oxford: Oxford University Press)

Gross C G 2002 Genealogy of the 'grandmother cell' *Neuroscientist* **8** 512–8

Grossberg S 2013 Adaptive resonance theory *Scholarpedia* **8** 1569

Haessig G, Cassidy A, Alvarez R, Benosman R and Orchard G 2018 Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system *IEEE Trans. Biomed. Circuits Syst.* **12** 860–70

Hammoodi M S, Stahl F and Badii A 2018 Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining *Knowl.-Based Syst.* **161** 205–39

Harnad S 1994 Preface *Mind Mach* **4** 377–8

He X, Liu T, Hadaeghi F and Jaeger H 2019b Reservoir transfer on analog neuromorphic hardware *Proc. 9th Int. IEEE/EMBS Conf. on Neural Engineering* pp 1234–8

He X, Sygnowski J, Galashov A, Rusu A A, Teh Y W and Pascanu R 2019a Task agnostic continual learning via meta learning *Proc. neurIPS 2019*

Hebb D O 1949 *The Organization of Behavior* (New York: Wiley)

Hinton G E, Osindero S and Teh Y-W 2006 A fast learning algorithm for deep belief nets *Neural Comput.* **18** 1527–54

Hinton G E and Salakuthdinov R R 2006 Reducing the dimensionality of data with neural networks *Science* **313** 504–7

Hodgkin A L and Huxley A F 1952 A quantitative description of membrane current and its application to conduction and excitation in nerve *J. Physiol.* **117** 500–44

Hoerzer G M, Legenstein R and Maass W 2014 Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning *Cerebr. Cortex* **24** 677–90

Hofstadter D 1995 *Fluid Concepts and Creative Analogies* (New York: Basic Books)

Hogan N and Flash T 1987 Moving gracefully: quantitative theories of motor coordination *Trends Neurosci.* **10** 170–4

Hopfield J J 1982 Neural networks and physical systems with emergent collective computational abilities *Proc. Natl Acad. Sci.* **79** 2554–8

Horsman D, Kendon V and Stepney S 2017 The natural science of computing *Commun. ACM* **60** 31–4

Howard M 2009 Computational models of memory *Encyclopedia of Neuroscience* vol 5 ed L Squire (London: Academic Press) pp 771–7

Huang C and Darwiche A 1994 Inference in belief networks: a procedural guide *Int. J. Approx. Reason.* **15** 225

Ielmini D and Wong H-S P 2018 In-memory computing with resistive switching devices *Nat. Electron.* **1** 333–43

Indiveri G *et al* 2011 Neuromorphic silicon neuron circuits *Front. Neurosci.* **5** 73

Indiveri G and Liu S-C 2015 Memory and information processing in neuromorphic systems *Proc. IEEE* **103** 1379–97

Izhikevich E M 2006 Polychronization: computation with spikes *Neural Comput.* **18** 245–82

Jaeger H 1996 Dynamische systeme in der Kognitionswissenschaft *Kognitionswissenschaft* **5** 151–74

Jaeger H 1998 Today's dynamical systems are too simple *Behav. Brain Sci.* **21** 643

Jaeger H 2000 Observable operator models for discrete stochastic time series *Neural Comput.* **12** 1371–98

Jaeger H 2007 Discovering multiscale dynamical features with hierarchical echo state networks *Technical Report 10* School of Engineering and Science, Jacobs University https://ai.rug.nl/minds/uploads/hierarchicalesn_techrep10.pdf

Jaeger H 2017 Using conceptors to manage neural long-term memories for temporal patterns *J. Mach. Learn. Res.* **18** 1–43 http://jmlr.org/papers/v18/15-449.html

Jaeger H 2019a *Theoretical Computer Science: I. Formal Languages and Logic. Online Lecture Notes* Jacobs University Bremen https://ai.rug.nl/minds/uploads/LN_FLL.pdf

Jaeger H 2019b *Theoretical Computer Science: II. Computability and Complexity. Online Lecture Notes* Jacobs University Bremen https://ai.rug.nl/minds/uploads/scriptCC.pdf

Jaeger H 2019c *Principles of Statistical Modeling. Lecture Notes* Jacobs University Bremen https://ai.rug.nl/minds/uploads/LN_PSM.pdf

Jaeger H, Doorakkers D, Lawrence C and Indiveri G 2021 Dimensions of timescales in neuromorphic computing systems *Deliverable Report for the European Union Project MeM-Scales*

James C D *et al* 2017 A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications *Biol. Inspired Cogn. Architect.* **19** 49–64

Jaynes E T 2003 *Probability Theory: The Logic of Science* (Cambridge: Cambridge University Press) first partial online editions in the late 1990s. First three chapters online at http://bayes.wustl.edu/etj/prob/book.pdf

Jordan M I 1996 Computational aspects of motor control and motor learning *Handbook of Perception and Action: Motor Skills* ed H Heuer and S Keele (New York: Academic)

Khansari-Zadeh S M and Billard A 2011 Learning stable nonlinear dynamical systems with Gaussian mixture models *IEEE Trans. Robot.* **27** 943–57

Kiebel S J, Daunizeau J and Friston K J 2008 A hierarchy of time-scales and the brain *PLoS Comput. Biol.* **4** e1000209

Kirkpatrick S, Gelatt C D and Vecchi M P 1983 Optimization by simulated annealing *Science* **220** 671–80

Kistler W M and Zeeuw C I D 2002 Dynamical working memory and timed responses: the role of reverberating loops in the olivo-cerebellar system *Neural Comput.* **14** 2597–626

Kloeden P E and Pötzsche C (ed) 2013 *Nonautonomous Dynamical Systems in the Life Sciences* (*Lecture Notes in Mathematics* vol 2102) (Cham: Springer)

Kloeden P E and Rasmussen M 2011 *Nonautonomous Dynamical Systems* (Providence, RI: American Mathematical Society)

Kuehn C 2015 *Multiple Time Scale Dynamics* (Springer)

Laird J E, Lebiere C and Rosenbloom P S 2017 A standard model of the mind: toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics *AI Mag.* **38** 13–26

Lakoff G 1987 *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind* (Chicago, IL: University of Chicago Press)

Lakoff G and Nunez R E 2000 *Where Mathematics Comes from: How the Embodied Mind Brings Mathematics into Being* (New York: Basic Books)

Land W M, Vochenkov D, Bläsing B E and Schack T 2013 From action representation to action execution: exploring the links between cognitive and biomechanical levels of motor control *Front. Comput. Neurosci.* **7** 127

Legenstein R, Papadimitriou C H, Vempala S and Maass W 2016 Assembly pointers for variable binding in networks of spiking neurons (arXiv:1611.03698)

Lenarz T 2017 Cochlear implant—state of the art *GMS Curr. Top. Otorhinolaryngol., Head Neck Surg.* **16** Doc04

Leonov A O and Mostovoy M 2015 Multiply periodic states and isolated skyrmions in an anisotropic frustrated magnet *Nat. Commun.* **6** 8275

Lins J and Schöner G 2014 Neural fields *Neural Fields: Theory and Applications* ed S Coombes, P beim Graben, R Potthast and J Wright (Berlin: Springer) pp 319–39

Littman M L, Sutton R S and Singh S 2001 Predictive representation of state *Advances in Neural Information Processing Systems 14 (Proc. NIPS 01)* 1555–61

Lloyd S 2013 The universe as quantum computer *A Computable Universe: Understanding and Exploring Nature as Computation* ed H Zenil (Singapore: World Scientific) pp 567–81

Lukoševičius M and Jaeger H 2009 Reservoir computing approaches to recurrent neural network training *Comput. Sci. Rev.* **3** 127–49

MacLennan B J 2004 Natural computation and non-turing models of computation *Theor. Comput. Sci.* **317** 115–45

Manjunath G and Jaeger H 2014 The dynamics of random difference equations is remodeled by closed relations *SIAM J. Math. Anal.* **46** 459–83

Markert H, Kaufmann U, Kara Kayikci Z and Palm G 2009 Neural associative memories for the integration of language, vision and action in an autonomous agent *Neural Netw.* **22** 134–43

Maturana H and Varela F J 1984 *The Tree of Knowledge: the Biological Roots of Human Understanding* (Boston, MA: Shamhala Press)

Mead C 1990 Neuromorphic electronic systems *Proc. IEEE* **78** 1629–36

Merolla P A *et al* 2014 A million spiking-neuron integrated circuit with a scalable communication network and interface *Science* **345** 668–73

Metropolis N, Rosenbluth A W, Rosenbluth M N, Teller A H and Teller E 1953 Equation of state calculations by fast computing machines *J. Chem. Phys.* **21** 1087–92

Mills J W 2008 The nature of extended analog computer *Physica*D **237** 1235–56

Mirigliano M, Decastri D, Pullia A, Dellasega D, Casu A, Falqui A and Milani P 2020 Complex electrical spiking activity in resistive switching nanostructured au two-terminal devices *Nanotechnology* **31** 234001

Moore C 1996 Recursion theory on the reals and continuous-time computation *Theor. Comput. Sci.* **162** 23–44

Moradi S, Qiao N, Stefanini F and Indiveri G 2018 A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps) *IEEE Trans. Biomed. Circuits Syst.* **12** 106–22

Mumford D 2002 Pattern theory: the mathematics of perception *Proc. ICM 2002* vol 1 pp 401–22

Mumford D 1994 Pattern theory: a unifying perspective *Proc. of 1st European Congress of Mathematics* ed A Joseph, F Mignot, F Murat, B Prum and R Rentschler (Basel: Birkhäuser) pp 187–224

Murmann B and Hoefflinger B (ed) 2020 *Nano-Chips 2030: On-Chip AI for an Efficient Data-Driven World* (Basel: Springer)

Murphy K P 2002 Dynamic Bayesian networks: representation, inference and learning *PhD Thesis* University of California, Berkeley

Murray J D *et al* 2014 A hierarchy of intrinsic timescales across primate cortex *Nat. Neurosci.* **17** 1661–3

Nakajima K, Fujii K, Negoro M, Mitarai K and Kitagawa M 2019 Boosting computational power through spatial multiplexing in quantum reservoir computing *Phys. Rev. Appl.* **11** 034021

Neal R M 1993 Probabilistic inference using Markov chain Monte Carlo methods *Technical Report CRG-TR-93-1* Department of Computer Science, University of Toronto

Neckar A *et al* 2019 Braindrop: a mixed-signal neuromorphic architecture with a dynamical systems-based programming model *Proc. IEEE* **107** 144–64

Newell A 1990 *Unified Theories of Cognition* (Cambridge, MA: Harvard University Press)

Newell A and Simon H A 1976 Computer science as empirical inquiry *Commun. ACM* **19** 113–26

O'Reilly R C, Braver T S and Cohen J D 1999 A biologically-based computational model of working memory *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control* ed A Miyake and P Shah (Cambridge: Cambridge University Press) pp 375–411

Olah C, Mordvintsev A and Schubert L 2017 Feature visualization *Distill* **2** e7

Olshausen B, Anderson C and Van Essen D 1993 A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information *J. Neurosci.* **13** 4700–19

Palm G 1980 On associative memory *Biol. Cybern.* **36** 19–31

Panzeri S, Harvey C D, Piasini E, Latham P E and Fellin T 2017 Cracking the neural code for sensory perception by combining statistics, intervention, and behavior *Neuron* **93** 491–507

Pascanu R and Jaeger H 2011 A neurodynamical model for working memory *Neural Netw.* **24** 199–207

Pecevski D, Buesing L and Maass W 2011 Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons *PLoS Comput. Biol.* **7** e1002294

Peitgen H-O 1986 *The Beauty of Fractals: Images of Complex Dynamical Systems* (Berlin: Springer)

Peláez E 1990 Parallelism and the crisis of von Neumann computing *Technol. Soc.* **12** 65–77

Pfeifer R and Scheier C 1999 *Understanding Intelligence* (Cambridge, MA: MIT Press)

Poetzsche C 2011 Nonautonomous bifurcation of bounded solutions: II. A shovel-bifurcation pattern *Discrete Continuous Dyn. Syst.* A **31** 941–73

Port R F and van Gelder T (ed) 1995 *Mind as Motion* (Cambridge, MA: MIT Press)

Preston A R and Eichenbaum H 2013 Interplay of hippocampus and prefrontal cortex in memory *Curr. Biol.* **23** R764–73

Prucnal P R, Shastri B J, Fischer I and Brunner D 2020 Introduction to JSTQE issue on photonics for deep learning and neural computing *IEEE J. Sel. Top. Quantum Electron.* **26** 0200103

Prychynenko D, Sitte M, Litzius K, Krüger B, Bourianoff G, Kläui M, Sinova J and Everschor-Sitte K 2018 Magnetic skyrmion as a nonlinear resistive element: a potential building block for reservoir computing *Phys. Rev. Appl.* **9** 014034

Rabe F 2008 Representing logics and logic translations *PhD Thesis* School of Engineering and Science, Jacobs University Bremenhttp://kwarc.info/frabe/Research/phdthesis.pdf

Rabinovich M I, Huerta R, Varona P and Afraimovich V S 2008 Transient cognitive dynamics, metastability, and decision making *PLoS Comput. Biol.* **4** e1000072

Ruiz Euler H-C, Alegre-Ibarra U, van de Ven B, Broersma H, Bobbert P A and van der Wiel W G 2020 Dopant network processing units: towards efficient neural-network emulators with high-capacity nanoelectronic nodes (arXiv:2007.12371)

Rumelhart D E and McClelland J L (ed) 1986 *Parallel Distributed Processing* vol 1 (Cambridge, MA: MIT Press)

Sabour S, Frosst N and Hinton G E Dynamic routing between capsules 2017 *Proc. NIPS 2017* http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf

Saunders D J, Patel D, Hazan H, Siegelmann H T and Kozma R 2019 Locally connected spiking neural networks for unsupervised feature learning *Neural Netw.* **119** 332–40

Schmidt S J (ed) 1987 *Der Diskurs des Radikalen Konstruktivismus* (Frankfurt: Suhrkamp Verlag)

Schöner G 2019 The dynamics of neural populations capture the laws of the mind *Top. Cogn. Sci.* **12** 1257

Schöner G, Haken H and Kelso J A S 1986 A stochastic theory of phase transitions in human hand movement *Biol. Cybern.* **53** 247–57

Shannon C E 1941 Mathematical theory of the differential analyzer *J. Math. Phys.* **20** 337–54

Shastri L 1999 Advances in shruti—a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony *Artif. Intell.* **11** 79–108

Shub M 2007 Morse–Smale systems *Scholarpedia* **2** 1785

Silver D *et al* 2016 Mastering the game of go with deep neural networks and tree search *Nature* **529** 484–9

Slotine J-J E and Lohmiller W 2001 Modularity, evolution, and the binding problem: a view from stability theory *Neural Netw.* **14** 137–45

Smith L B and Thelen E (ed) 1993 *A Dynamic Systems Approach to Development: Applications* (Cambridge, MA: MIT Press)

Smolensky P 1990 Tensor product variable binding and the representation of symbolic structures in connectionist systems *Artif. Intell.* **46** 159–216

Steels L, Steels L and Brooks R A 1993 Building Agents out of Autonomous Behavior Systems *The Artificial Life Route to Artificial Intelligence* (Milton Park: Routledge)

Stepney S and Hickinbotham S 2018 UCOMP roadmap: survey, challenges, recommendations *Computational Matter* ed S Stepney, S Rasmussen and M Amos (Basel: Springer) ch 1 pp 9–32

Stepney S and Kendon V 2020 The representational entity in physical computing *Nat. Comput.* (accepted)

Stepney S 2017 Introduction to unconventional computing *Guide to Unconventional Computing for Music* ed E R Miranda (Basel: Springer) ch 1 pp 1–21

Stepney S, Rasmussen S and Amos M (ed) 2018 *Computational Matter* (*Natural Computing Series*) (Basel: Springer)

Strogatz S H 1994 *Nonlinear Dynamics and Chaos* (Reading, MA: Addison-Wesley Developers Press)

Sussillo D and Barak O 2013 Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks *Neural Comput.* **25** 626–49

Sutskever I, Hinton G E and Taylor G W 2009 The recurrent temporal restricted Boltzmann machine *Advances in Neural Information Processing Systems 21 (NIPS 08)* ed D Koller, D Schuurmans, Y Bengio and L Bottou pp 1601–8

Tanaka G, Yamane T, Héroux J B, Nakane R, Kanazawa N, Takeda S, Numata H, Nakano D and Hirose A 2019 Recent advances in physical reservoir computing: a review *Neural Netw.* **115** 100–23

Tarski A 2002 On the concept of following logically *Hist. Phil. Logic* **23** 155–96

Tenenbaum J B, Griffiths T L and Kemp C 2006 Theory-based Bayesian models of inductive learning and reasoning *Trends Cogn. Sci.* **10** 309–18

Tervo D G R, Tenenbaum J B and Gershman S J 2016 Toward the neural implementation of structure learning *Curr. Opin. Neurobiol.* **37** 99–105

Tetzlaff C, Kolodziejski C, Markelic I and Wörgötter F 2012 Time scales of memory, learning, and plasticity *Biol. Cybern.* **106** 715–26

Thoroughman K A and Shadmehr R 2000 Learning of action through adaptive combination of motor primitives *Nature* **407** 742–7

Thorpe S, Delorme A and Van Rullen R 2001 Spike-based strategies for rapid processing *Neural Netw.* **14** 715–25

Thrun S *et al* 2006 Stanley: the robot that won the DARPA grand challenge *J. Field Robot.* **23** 661–92

Tino P, Horne B G and Giles C L 1998 Finite state machines and recurrent neural networks—automata and dynamical systems approaches *Neural Networks and Pattern Recognition* ed O Omidvar and J Dayhoff (New York: Academic) ch 6 pp 171–219

Torrejon J *et al* 2017 Neuromorphic computing with nanoscale spintronic oscillators *Nature* **547** 428–31

Tsuda I 2001 Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems *Behav. Brain Sci.* **24** 793–810

Turing A M 1936 On computable numbers, with an application to the Entscheidungsproblem *Proc. Math. Soc.* **42** 230–65

van Gelder T and Port R (ed) 1995 *Mind as Motion: Explorations in the Dynamics of Cognition* (Cambridge, MA: MIT Press)

van Leeuwen J and Wiedermann J 2001 Beyond the turing limit: evolving interactive systems *Int. Conf. on Current Trends in Theory and Practice of Computer Science* (Springer) 90–109

van Noort D, Gast F-U and McCaskill J S 2002 DNA computing in microreactors *7th Int. Workshop on DNA-Based Computers, DNA7* ed N Jonoska and N C Seeman (Springer) pp 33–45

Varela F G, Maturana H R and Uribe R 1974 Autopoiesis: the organization of living systems, its characterization and a model *Biosystems* **5** 187–96

von Bertalanffy L 1968 *General System Theory* (New York: Braziller)

von Neumann J 1956 Probabilistic logics and the synthesis of reliable organisms from unreliable components *Automata Stud.* **34** 43–98

von Neumann J and Burks A W 1966 *Theory of Self-Replicating Automata* (Champaign, IL: University of Illinois Press)

Wainwright M J and Jordan M I 2003 Graphical models, exponential families, and variational inference *Technical Report 649* BerkeleyDe-partment of Statistics, University of California

Waldrop M M 2016 The chips are down for Moore's law *Nature* **530** 144–7

Wernecke H, Sándor B and Gros C 2018 Attractor metadynamics in terms of target points in slow–fast systems: adiabatic versus symmetry protected flow in a recurrent neural network *J. Phys. Commun.* **2** 095008

Wiener N 1948 *Cybernetics, or Control and Communication in the Animal and the Machine* (Cambridge, MA: MIT Press)

Wilson R A and Foglia L 2017 Embodied cognition *The Stanford Encyclopedia of Philosophy* ed E N Zalta (Stanford: Metaphysics Research Lab, Stanford University)

Wiskott L and Sejnowski T J 2002 Slow feature analysis: unsupervised learning of invariances *Neural Comput.* **14** 715–70

Wolfram S 2002 *A New Kind of Science* (Champaign, IL: Wolfram Media)

Wolpert D H 2015 Extending Landauer's bound from bit erasure to arbitrary computation (arXiv:1508.05319)

Wunsch G 1985 *Geschichte der Systemtheorie* (München: Oldenbourg)

Yamashita Y and Tani J 2008 Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment *PLoS Comput. Biol.* **4** e1000220

Yang J J, Strukov D B and Stewart D R 2013 Memristive devices for computing *Nat. Nanotech.* **8** 13–24

Yao Y and Freeman W J 1990 Model of biological pattern recognition with spatially chaotic dynamics *Neural Netw.* **3** 153–70

Yousefzadeh A, Stromatias E, Soto M, Serrano-Gotarredona T and Linares-Barranco B 2018 On practical issues for stochastic STDP hardware with 1-bit synaptic weights *Front. Neurosci.* **12** 665

Zauner K P 2005 From prescriptive programming of solid-state devices to orchestrated self-organisation of informed matter *Unconventional Programming Paradigms* ed J-P Banatre *et al* (Berlin: Springer) pp 47–55

Zhang H, Goodfellow I, Metaxas D and Odena A 2019 Self-attention generative adversarial networks *Proc. 36th Int. Conf. on Machine Learning* ed K Chaudhuri and R Salakhutdinov (California: Long Beach) pp 7354–63

Zhang Y *et al* 2020 A system hierarchy for brain-inspired computing *Nature* **586** 378–84

Zhao H, Waughray D, Malone D M, Msuya J, Ryder G, Bakker P, Seth N, Yong L and Payet R 2019 A new circular vision for electronics: time for a global reboot *Report in Support of the United Nations E-Waste Coalition* World Economic Forum https://weforum.org/reports/a-new-circular-vision-for-electronics-time-for-a-global-reboot

Zheng N and Mazumber P 2020 *Learning in Energy-Efficient Neuromorphic Computing: Algorithm and Architecture Codesign* (New York: Wiley)

Zuse K 1982 The computing universe *Int. J. Theor. Phys.* **21** 589