VRIJE UNIVERSITEIT BRUSSEL

DOCTORAL THESIS

Synthetic Cognitive Development of decentralized self-organizing systems

Author:
Viktoras Kabir VEITAS

Supervisor:
Prof. Dr. Francis HEYLIGHEN

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Interdisciplinary Studies

in the

ECCO – Center Leo Apostel
Doctoral School of Human Sciences

June 2019

Vrije Universiteit Brussel

Abstract

Faculty of Arts and Philosophy

Doctoral School of Human Sciences

Doctor of Philosophy in Interdisciplinary Studies

Synthetic Cognitive Development of decentralized self-organizing systems

by Viktoras Kabir VEITAS

This thesis is an interdisciplinary design inquiry into the operation of intelligence. It is aimed at conceiving a computational model of individuation of an open cognitive system and builds upon the evolutionary systemic framework of *open-ended intelligence*, which treats difference primal to identity, becoming primal to being, change primal to stability and communication primal to object. First, we devise the principles of the open cognitive system from selected concepts, techniques and currents of theoretical and pragmatic thinking in the domain of the evolution of mind, brain and body. Then, using the metaphysical framework of open-ended intelligence, these principles and requirements are integrated into a model of *synthetic cognitive development* which rests on the mechanism of progressive determination of systemic constraints in an evolutionary developmental way. Further, we formulate the computational perspective of the stigmergic cooperation of a population of independent and heterogeneous actors in terms of an *open-ended decentralized computing model*. Finally, we specify the semantics of the computational model and software design by integrating the actor model with graph computing and, through computational simulation experiments, demonstrate the stigmergic computing in the domain of decentralized exchange. The open-ended decentralized computing model proposes a path for conceiving, designing, simulating and engineering open systems by emphasizing the process of self-organization of an unconstrained space of possibilities. It extends the paradigmatic shift of open-ended intelligence from identity to individuation into the domain of engineering, particularly of artificial general intelligence and the ambition to integrate advanced autonomous technologies into the fabric of society.

Keywords: *Evolution, cognitive system, open-ended intelligence, individuation, artificial general intelligence, stigmergy, self-organization, decentralization, actor, graph, computational model*

Acknowledgements

I thank my supervisor Professor Dr. Francis Heylighen for his decades long academic presence in complexity and interdisciplinary science domain, which drew me to engage into this journey. I am deeply grateful for his encyclopaedic knowledge, incredible openness for discussion, advice, encouragement and allowance for exceptional freedom of deep intellectual exploration.

I would like to extend special gratitude to Dr. David R. Weinbaum (Weaver), member of my advisory and examining committees, for inviting me to the world of living philosophy of individuation and becoming, which became the cornerstone, guidance and research avenue for this work and beyond.

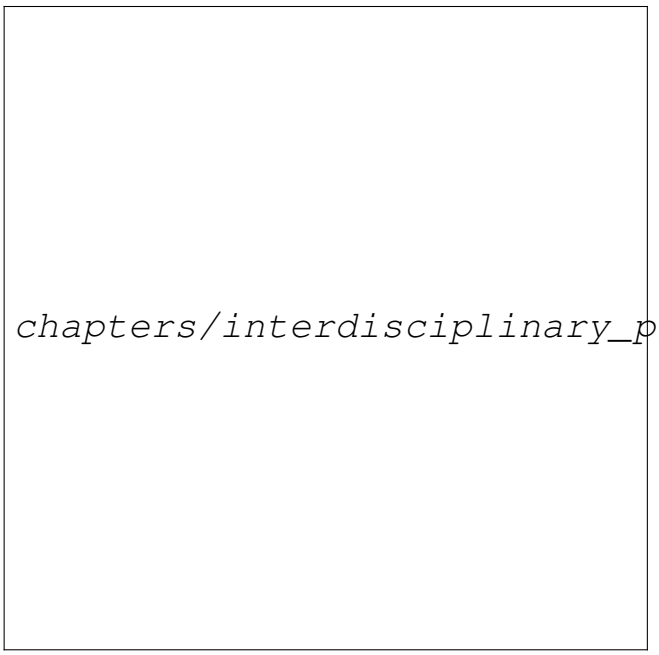
I express my sincere appreciation to the members of the examining committee Professor Ann Nowé (VUB), Professor Pieter Ballon (VUB), Dr. Marta Lenartowicz (VUB), Dr. Ben Goertzel (Xiamen University of Technology) and Dr. Harry Halpin (Massachusetts Institute of Technology).

I am grateful to Yuri Milner Foundation and SingularityNET Foundation for the funding and support. Many thanks to Ailsa Campbell for proofreading and grammar advice; to all members of the Global Brain Institute for the intellectual environment and eyes-opening, mind-bending discussions; most importantly – to all my friends for expressed or unspoken patience and love.

Contents

List of Figures

List of Tables



to No-body

Chapter 1

Framing an interdisciplinary problematique

I must create a system, or be enslaved by another person's. I will not reason and compare: my business is to create.

William Blake, poet

Those who are skilled in producing surprises will win.

Sun Tzu, military strategist

[..] focus at the intersection of theory and practice. There is no progress without friction.

Erik Meijer, computer scientist

1.1 Methodological approach

This work presents an interdisciplinary design inquiry into the operation of intelligence. As *interdisciplinary*, it builds on knowledge from numerous domains of science, including philosophy, artificial intelligence, computer, cognitive, and social sciences, as well as practical know-how in fields such as information system management. As a *design inquiry* it aims at creating an intelligent system rather than "just" formulating a theory with claims for generality or objectivity.

To my great surprise and wonder, the journey of the inquiry has led to immersion in the most abstract philosophical frameworks that I could have imagined and has contributed to the formulation of a conceptual approach to a synthetic and natural

intelligence in terms of *open-ended intelligence* and *individuation of intelligence* concepts. This approach has guided the whole inquiry, which has also been continuously shaped, reformulated and adjusted by the process.

Any interdisciplinary inquiry by definition dissolves one or more boundaries among scientific disciplines, modes of thinking and approaches. Therefore, it naturally calls for a certain amount of necessary conceptualisation for the internal logic of investigation. The broader the inquiry, the more theoretic conceptualisation is needed in order to keep its integrity. The conceptual framework of the present inquiry is ultimately broad, with its roots in the philosophy of becoming and individuation and the theory of evolution. Yet as much as we are tempted to formulate the "concept of intelligence" , it only makes sense in terms of concrete phenomena, to which it is inseparably connected – intelligent beings embodied in concrete bodies and embedded in a concrete environment.

Hence, the choice of a design-based research methodology for this work is motivated by my conviction that an investigation of intelligence must be approached not only via an interdisciplinary assemblage of concepts, principles and tools drawn from various disciplines, but, even more importantly, through interweaving them into a single thread – connecting ultimately abstract and broad to the most specific and vice versa. Needless to say this thread of thought should cut through numerous intermediate levels of abstraction. As a cotton thread immersed into a jar of salty water acts as an attractor for ions and their solidification into a column of crystals, the ambition of synthetic intelligence design provides a basis for organizing a rich solution of interdisciplinary knowledge about cognitive living systems.

A few important methodological considerations can be formulated with the help of this metaphor. First, the crystallization is a process of growth. This process, rather than exact configuration of the resulting crystal, is the locus of attention and interest. Likewise, in a design inquiry, assembling the interdisciplinary knowledge into an image of an intelligent system is an iterative process without a clear success criterion. Second, the exact result of a design inquiry is never known in advance, just as the precise macro structure of the crystal is not determined by putting a thread into a solution. Third, such a process has no determined end state, as a crystal stops growing only because the thread is pulled out of the solution by the hand of a player (considering that the pool of interdisciplinary knowledge is inexhaustible, contrary to the salt ions in a jar). The metaphor of crystallization goes deeper than illustrating the research methodology of this thesis – it is a metaphor for a general process of individuation and open-ended intelligence.

1.2 A dangerous method

As the metaphor of crystallization partially illustrates, a design-based interdisciplinary research is necessarily accompanied by a complex and therefore often inconvenient context, which needs to be taken into account when engaging and proceeding with the inquiry. Importantly, this context is also relevant to interdisciplinary research in general.

If taken at face value and without much consideration of the context, the statement "I will not reason and compare: my business is to create", chosen for the epigraph of the current chapter, may sound unscientific or even utterly unreasonable –

surely in science we have to reason and compare different approaches, possibilities, solutions and models. Yet I chose this quote to illustrate the self-reinforcing nature of evolutionary process that cognition is – emphasizing the serendipitous “considerations” and “choices” that a process takes without supporting them with evidence – simply because no evidence exists before a process of cognition starts (Weinbaum, 2018).

Interdisciplinary investigation presents the dilemma of the fine balance in choosing between, on the one hand, broad coverage of many domains and ways of thinking, and, on the other, a deep investigation of each issue – neither of which is actually possible due to constraints of time and space. None of these modes of thinking is inherently better or worse than another: it is easy to overlook important details when taking a “generalist” view; but it is as easy to lose high-level understanding of an issue due to digging too deep into details and specifics. We attempt to deal with this dilemma by utilizing the principle which states that one should enter into the details of each branch of investigation as deeply – but not more – as needed to justify a chosen direction of further design inquiry from that point. Applying this principle is a somewhat frustrating experience as it often involves a fair amount of uncertainty over deciding to draw interim conclusions without investigating the issue "to the end". This uncertainty is decreased by iteratively revisiting interim decisions and considerations at every loop of the design inquiry, which brings about another, rather technical, inconvenience – no stage of the design process (i.e. chapter of the thesis in this case) can be considered finished unless the whole process is "finished".

The work persistently negates the primacy of top-down and bottom-up approaches over one another and holds that both modes of thinking need to be used simultaneously, or, at worst, iteratively. Design and management practitioners and scientists as well as software and hardware engineers face the issue of combining the top-down and bottom-up thinking every day. They have developed a plethora of methods and tools for dealing with it – under the labels of action research, iterative design, test-operate-test-exit loop, orient-observe-decide-act loop, fast prototyping, agile methodology and many more. All these methods build upon the same idea of establishing a "top-bottom-top-bottom-top" design loop and iterating it as fast as possible – in this way trying to capture both perspectives at the same time.

An important technique helps in navigating this fundamentally open-ended process of the design inquiry in this work. The technique is the very action of emphasizing the "mechanistic" aspect of the process of cognition and intelligence – i.e. looking at the living and intelligent complex system as a type of machine, which, being much less deterministic than we usually attribute to the word, can still be simulated using mathematical and computational methods. By this, we link philosophical concepts of open-ended intelligence and individuation of intelligence to its concrete and possible manifestations.

1.3 A precarious landscape of inquiry

With a certain irony, we can see how the mechanistic aspect actually links images of "human – biological – natural" to those of "synthetic – artificial – constructed" intelligence, instead of, as so often perceived, differentiating between them. The irony lies in the observation that, while many schools of thought implicitly or explicitly posit irreducibility of the phenomenon of cognition to a physical process (using images of,

e.g., soul or else), at the same time cultures based on these schools of thought routinely treat humans as machines. The property of being human is often measured by the performance of individuals according to their social role as citizens, soldiers, good husbands or wives, dedicated employees and the like. It seems to me that this is precisely why the idea that "AI will take over the world and exterminate humans"[1] resonates so well in some (mostly "western") – societies. Partially it is because the concept of intelligence is continuously being reduced to the efficient optimization function of a behaviour in a particular environment (Bostrom, 2012), which AI is poised to perform better. I believe that intelligence is much more than optimization – it is an open-ended creative exploration for opportunities and growth via constant re-definition of itself. The artificial intelligence programme – the quest to create a synthetic mind – is a fertile context of exploring the open-endedness of intelligence in the broadest possible sense.

I believe that there are many reasons why the collective psyche has developed such inconsistent patterns of making sense of "human nature" and intelligence at large. One of the reasons which I find fundamental and particularly interesting in terms of this work has to do with the emphasis and desire for **predictability of the world**. It is natural for brains and minds to search for repeating patterns in environmental stimuli and infer causal relationships between them. This process lies at the core of making sense of the world, everything around and ourselves – hence intelligence. Almost always we interpret sense-making as a some sort of asymptotic optimization process aimed at acquiring the “true” or “best” picture of the world, “correct” thinking and “useful” behavioural patterns. This is a presumption that has proved to be of immense value for the development of the human mind, culture and science, yet is precisely the one that I would like to challenge – not because it is not “true”, but because I believe it to be a special case of a broader process of individuation of intelligence, which is divergent rather than convergent. In this thesis the divergent aspect of the process of intelligence is investigated as manifested in decentralized computing and the reflexive nature of distributed intelligence and social systems.

Usually, making sense of anything is related to finding some order within disorder – so constructing a model of reality and exposing it to rigorous checking and testing against that reality. This is the so-called hypothetico-deductive method of science. A deep underlying assumption behind the method is that there is an order that can be found. So thinking in this case starts from assuming the order, devising a model of it and then augmenting and changing it, if it does not stand up to whatever tests the world presents. Another way of thinking, which we are embracing and advocating here – without denying the hypothetico-deductive one – is starting from the assumption of disorder and bringing order to it by devising models and explanations. It may sound like a subtle nuance, yet conceptually it makes all the difference in the world – as it introduces the need to "to put into the driver’s seat" a subjective perspective of an entity that makes sense of the world. The central question becomes not “what is order or what is disorder” but how order comes about from disorder. In other words – how the process of *self-organization* happens – first and foremost in the perceiver’s mind. For the purposes of this research it means that we are mostly attending to the question of *how* models and images of the world are devised that make sense in that world for a particular subject rather than *what* the true or correct models are. If we translate this principle to the quest for artificial

[1] https://en.wikipedia.org/wiki/AI_takeover

general intelligence, the resulting architecture should encompass a mechanism for creating images of the world by making sense of the unknown and unknowable – via embodying open-ended intelligence principles.

The prevailing emphasis on asymptotic processes has peculiar spillover effects upon other concepts. In particular, the concepts of "computing" and "computation" are often by default considered predictable and deterministic processes. Of course, the focus on deterministic computation (clearly related to the desire for predictability of the world) is well understandable since why would anyone want to start a computational process which was not known to lead to "useful" results when criteria for "usefulness" could be defined before starting a process? Even so, it is first of all consequential to note that deterministic computation is a special case of non-deterministic computation[2], and therefore the concept of computation covers both. Second, there is at least one large family of systems (*complex adaptive systems* – which cannot be understood by breaking them into parts) and related processes (*self-organization* of structure and patterns without external control) that are inherently non-deterministic. An important corollary of this thesis is that the essentials of complex adaptive systems and self-organization are more important for operation and development of society, intelligence, cognition, living systems and evolution at large than any deterministic (or even probabilistic) processes observable with respect to them. I hope at least to show that it is a valid perspective to consider – but of course aim at convincing the reader that it is a much richer perspective, having practical importance when designing and operating artificial systems exhibiting increasing levels of autonomy and intelligence. This does not deny the value of understanding and designing predictable and deterministic processes – once again considering that they are special cases of non-determinism. The questions of interest are then: how does an inherently non-deterministic process start to behave predictably, or how can it be modelled as such? How do the constraints of such process come to be from nothing and how can we formulate all this in computational terms?

With respect to the domain of social science, I am of the opinion that a shift in the perspective towards considering a divergent nature of intelligence, cognition, social development and evolution at large is of more relevance than ever when the exponential growth of technology and human population has accumulated a momentum powerful enough to pivot dynamics of the whole system. This relevance is clearly apparent in the climate change issue: it is clear that humanity is no longer able conveniently to assume that it is not responsible for the reality (environmental and otherwise) it lives in. My interest with respect to this example is to show how the prevalent modes of thinking guide and largely constrain the space of future possibilities that we see, reflect upon and actuate – reflexively shaping the reality in a way comparable to how self-fulfilling prophecies come "true".

1.4 Structure of the thesis

The thesis is loosely structured into three interrelated parts – dealing with the conceptual aspect (Chapters 2 and 3), computational aspect (Chapters 4 and 5) and domains of application (Chapters 6 and 7). Chapter 2 discusses relevant existing and historical perspectives to AI research, evolution of brain, body and mind with

[2]See debate about deterministic vs. non-deterministic Turing machines in Section 4.2.3 Deterministic versus non-deterministic computation on page 90.

insights from cognitive and neuro-science. Chapter 3 presents the open-ended intelligence philosophy and its roots in the philosophy of individuation, and positions both historical perspectives and interdisciplinary approaches towards evolution of life and mind within it. In Chapter 4 we investigate the computational aspects of open-ended intelligence and develop a model of open-ended decentralized computing. Chapter 5 proposes and discusses concrete models for implementing open-ended decentralized computing and proposes the architecture that integrates two paradigms – the actor model of computation and graph computing. Chapter 6 presents an example of preliminary implementation of the software architecture of open-ended computing, utilizing the concept of decentralized exchange. Chapter 7 discusses broad domains of applicability of the open-ended decentralized computing model and other prospective avenues of its implementation. Finally, in Chapter 8 we look at what came out from "keeping a thread in a jar of salty water" in terms of integration of conceptual, computational and software development perspectives. Further, in terms of future research, the relevance of the open-ended decentralized computing model for dealing with socio-technological challenges and fostering advances of the human society are discussed. With this structure we attempt to expound the highly interrelated material in a linear manner. This is possible only partially, since the value of interdisciplinary research comes first and foremost from interconnectedness of knowledge from different disciplines, whereas these connections are largely cyclical. I have made my best efforts to introduce earlier concepts that are used later. Still, since the relations between these concepts are in large part the content of this work, the text includes many internal backward and forward references. To best fully appreciate these relations it is advisable to read at least Chapters 2, 3 and 4 straight through and only then follow the internal references. Additionally, a short glossary at the end of this chapter aims to introduce and partially describe the main concepts and relations.

A few more technical issues before we proceed. First, I purposefully use first person pronoun "I" and third person pronoun "we" interchangeably throughout the text. The default pronoun is "we", which reflects the approach that ideas do not have a single source and emphasizes the "decentralization" of the thinking process as well as inviting the reader to participate in it. The pronoun "I" is used for denoting my more subjective intuitions, perspectives and images. Second, double quotation marks are used both for denoting quotes of other people and a figurative speech. Third, both *italic* and **bold** text is used for denoting definitions, to draw attention or simply accentuate words and sentences without following any more strict rules of using them.

1.5 Glossary

This section provides a short glossary of the main concepts and their relations used in the thesis. Descriptions below aim to provide high-level introductions rather than theoretically rigorous definitions. Full definitions within their respective contexts are developed further in the text.

Individuation is the process of formation of individuals. It is a central concept of the philosophy of individuation by Simondon (1992, 2005) which provides a shift of perspective from individuals as the primary elements of reality and

the world to the process of becoming which brings fourth those individuals to being.

Progressive determination is a mechanism explaining how constraints that guide individuation and development of a system progressively emerge within the very same process of development – creating and progressively constraining the "space of possibilities" of further development. Formally, the mechanism of progressive determination can be represented as a chain of transformations where an operation transforms structure and structure in turn transforms operation. The major contribution of this thesis is the computational representation of the progressive determination.

Assemblage is a sub-network of individuals that have established partial compatibility among themselves operating within a larger population of interacting individuals. An assemblage possesses an intrinsic though metastable individuality on its own, over and above individualities of its members. Assemblages get formed via the mechanism of progressive determination when emerging patterns of interaction and compatibility between its members constrain further development trajectory of the individuality of an assemblage.

Scales of individuation is the recursive structure of assemblages within assemblages. It is a concept that describes assemblages acting as individuals and forming further assemblages at a higher scale of recursive hierarchical structure. This can be illustrated in terms of a biological metaphor. Cells are individuals that get assembled in to organs, which are individuals that get assembled into organisms. Organisms are then individuals that assemble themselves into groups or societies with their own distinct individuality and collective intelligence (as, e.g. bee hives or anthills).

Pre-individual, fluid individual and fully-formed individual refer to assemblages of different consolidation levels. **Pre-individual** is a population of disparate individuals which are incompatible between themselves and therefore no assemblages can be observed or identified in it. Pre-individual contains in itself open-ended possibilities of development, since no developmental trajectories can yet be identified. A **fluid individual** is an observable assemblage which nevertheless does not have a clearly consolidated boundary or agency. A **fully-formed individual** is then an assemblage with clear boundaries and consolidated agency. The mechanism of progressive determination describes the gradual emergence of fluid and fully-formed individuals from a pre-individual.

Open-ended intelligence is the individuation of intelligence and its manifestations via the mechanism of progressive determination. It is a process where a distributed population of interacting heterogeneous agents achieves progressively higher levels of coordination and form assemblages possessing agency and intelligence. Coordination amounts to the local resolution of disparities by means of reciprocal determination that brings forth new individuals in the form of integrated groups of agents (assemblages) that exchange meaningful information and spontaneously differentiate (dynamically and structurally) from their surrounding milieu (Weinbaum and Veitas, 2017b).

Goal-directed intelligence is the term that defines intelligence as a set of skills, abilities and efficient optimization processes for reaching a well defined set of

goals in a particular environment. Goal-directed intelligence underlies the prevailing mode of thinking about intelligence of organisms which are necessarily embodied and embedded in certain environments and therefore manifest goal-directed behaviours. Open-ended intelligence is the process of emergence of intelligence itself, including goal-directed intelligences and their manifestations.

Artificial General Intelligence (AGI) is the program for researching, understanding and engineering intelligent systems which could perform any intellectual task that a human can and eventually beyond. AGI is considered a return to the original AI problem of "how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves" (McCarthy et al., 2006). In contrast, **narrow AI** – a somewhat more known perspective towards artificial intelligence – is concerned with isolated abilities, skills and properties of intelligence. The relation between narrow and general AI in this work is approached through the concept of open-ended intelligence – an overarching philosophy of intelligence, biological and artificial alike.

Stigmergy is a form of indirect coordination between independent actors via a shared medium, where some actors leave a trace that is picked up and acted upon by other actors and in such manner guides their actions. It is probably the simplest yet most effective form of coordination of complex systems – so effective that it gives rise to the phenomenon of *collective intelligence* famously observed in large colonies of eusocial insects, where a group is observably more intelligent than any single individual in it. The notion of **stigmergic computing** refers to generalized stigmergic cooperation as a model of computing.

Selection for relevance is an attention mechanism which accounts for a choice to attend to certain aspects of the environment and its signals while relatively downplaying others. Selection for relevance in cognition allows to make sense of unstructured and rich environments by a cognitive system operating within constraints of limited resources of space and time.

Reflexivity refers to the circular relationships between cause and effect when each element both affects and is affected by other elements. In particular, it refers to a feedback relationship between observer (i.e. intelligent agent) and observed (i.e. environment): any examination and action of agents “bend” the environment and affect the perception and further decisions by the same agents. Reflexivity underlies the interaction among scales of individuation.

Synthetic cognitive development is a model of individuation of cognition from the vantage point of an individuating agent and is a generalization of human cognitive development to any intelligent system. It operates within the framework of scales of individuation utilizing the mechanism of progressive determination.

Actor model is an established mathematical model of concurrent computation. Actors in a model communicate via messages each of which constitutes a communication event. Computation happens when events get ordered. Actor model constitutes a single currently viable super-Turing computing model.

Graph computing is a quickly developing new paradigm of computing which leverages network science and graph theory to represent data, programs and their

interactions in distributed computing environments. It is a large set of technologies and a way of modelling the world in terms of entities (nodes) and their relations (semantic or analogous associations, causality, information and control flows).

Graph traversal is a central concept in graph computing and is a systematic method of exploring vertices and edges in a graph. Traversals are represented using special purpose *graph traversal languages* which formalize an abstract description of a legal path through a graph. Graph traversing is then a process of visiting (checking, updating or modifying) vertices and links of a graph, based on the *imperatively* defined constraints by a user.

Open-ended decentralized computing model is the computational representation of the process of progressive determination. Its architecture unites the Actor model for representing decentralized communicating processes with graph computing for representing coordination between processes in terms of persistent topological relations between them. Open-ended decentralized computing model represents the mechanism of progressive determination in terms of evolutionary development of ordered computation graphs in an initially disordered population of elementary processes.

Chapter 2

A quest to understand

2.1 AI research perspectives

The modern quest for creating machines capable of thinking like humans began around the middle of the twentieth century, undoubtedly triggered by the advent of digital computers. Two major events are usually mentioned as indicating the beginning of the artificial intelligence (AI) research programme: Alan Turing's article "Computing Machinery and Intelligence" (Turing, 1950) and the Dartmouth Summer Research Project on Artificial Intelligence of 1956.

In the course of six decades of research and scientific and popular discourse around AI, a number of informal and formal descriptive terms have emerged, aimed at indicating different aspects or types of intelligence, as well as research perspectives. We review the major concepts and the contexts in which they are used in order to pave the way for further discussion and illustrate the precariousness and controversy around the concept of intelligence at large.

2.1.1 General artificial intelligence

General artificial intelligence (AGI) is considered by its proponents a return to the roots (Goertzel, Pennachin, and Geisweiller, 2014) of the original AI research programme formulated by the organizers of the Dartmouth Workshop in 1956 as "an attempt [...] to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves" (McCarthy et al., 2006).

With the aim of coming up with a standard definition of *universal intelligence*, Legg and Hutter (2007) collected over 70 definitions of intelligence and differentiated them into three broad categories: (1) collective (found in encyclopaedias and dictionaries), (2) psychologist and (3) AI researcher definitions. They then distilled their own account – ***intelligence measures an agent's ability to achieve goals in a wide range of environments*** – by observing the most common features used for describing intelligence:

- *embodiment*: a property that an individual agent has as it interacts with its environment or environments;
- *goal directness*: related to the agent's ability to succeed or profit with respect to some goal or objective;
- *efficiency*: dependence on how able the agent is to adapt to different objectives and environments.

Likewise, Goertzel (2009) proposes a notion of *efficient pragmatic general intelligence* and defines it as ***the capability of a system to choose actions maximizing its goal-achievement, based on its perceptions and memories, and making reasonably efficient use of its computational resources***.

A somewhat less pragmatic, but still computationally expressible definition is iterated by Battaglia et al. (2018) with relation to linguistic theory, referencing Chomsky (1969) and Humboldt (1988), that intelligence is the ***infinite use of finite means*** in which a small set of elements can be productively combined in limitless ways.

2.1.2 Narrow artificial intelligence

As defined by Kurzweil (2005), narrow AI refers to machines and algorithms which perform a specific function that once required human intelligence to perform, and does it at human levels or better. The difference between the concepts of "general" and "narrow" AI seems to be "only" that the latter emphasizes different functions considered intelligent, while the former emphasizes the integration of these functions into a single system capable of more than the sum of its parts. Interestingly, the definition of "narrow" AI is mostly used by AGI researchers to distinguish their research programme from the multitude of research areas in machine perception, natural language processing, machine learning, sensor fusion, neural networks, and many others, which, for arguably historical reasons, have been placed under the umbrella of AI research (Goertzel, Pennachin, and Geisweiller, 2014).

Yet, while the historical separation is obvious, the conceptual borderline is not that straightforward, especially considering the latest achievements of the "narrow" AI[1]. As a research field, "narrow" AI holds the potential and ambition to reach the original AI agenda, albeit this has started to be vocalized only lately. For example, Jürgen Schmidhuber, one of the pioneers of "deep learning" techniques, powering much of the current success of "narrow" AI, envisions how artificial general intelligence could grow out of current specialized pattern recognition networks using the principles of reinforcement learning (Kahrs, 2017).

2.1.3 Global brain

The *global brain* is a metaphor for an emerging intelligent network that is formed by all people with computers, knowledge bases and communication links that connect them together (Heylighen, 2002b). Following the metaphor, the global brain is the nervous system of the organism of human society. The origin of the concept can be tracked at least to the middle of the twentieth century, yet gained a scientific and

[1] AlphaGo – an AI Go player; Libratus – an AI Poker player

technical perspective with the rise of computer networks, the internet and social networks.

Apart from social, organismic, philosophical, utopian, technical, cybernetic and many other perspectives, the global brain metaphor first of all emphasizes the concept of *distributed intelligence*, which emerges from a network of interacting heterogeneous agents of lower capabilities. The relation of intelligence and network structure is strongly grounded in neurophysiology and is very well reflected in AI research and practical applications. A network of interconnected and interacting processes is largely regarded as a correct image of an intelligence machinery by pioneer as well as contemporary AI researchers (Goertzel, 2002; Minsky, 1988) and forms the basis of the *connectionist* approach.

Relations in a network naturally represent interactions and interrelations among many heterogeneous agents that are dynamic, seemingly chaotic, evolving, and do not fit into clear logical and semantic structures. Yet the main message of the global brain metaphor lies in its emphasis on the importance of *self-organized coordination of decentralized processes*. Actually, all intelligence is essentially decentralized and distributed, albeit to a different degree: brains and minds are products of interactions of neurons and coordination of neural activity in brain areas; collective intelligence of eusocial insects originates from their interaction via pheromone trails; the intelligence of civilizations and societies as well as of companies and organizations emerges from coordination among individual humans. In terms of the conceptual perspective that we embrace in this work, the fundamental mechanisms of intelligence transcend boundaries of these concrete embodiments and forms. We therefore consider artificial general intelligence, global brain research and distributed intelligence all as aspects of the same quest for understanding intelligence at large – mutually informing and enforcing rather than contradicting each other.

2.1.4 Universal intelligence

The theory of universal intelligence provides a formal definition of a universally intelligent agent (AIXI), able "to achieve goals in a wide range of environments", which conforms to the definition of artificial general intelligence given above. Direct practical application of this theoretical and mathematical abstraction of an intelligent agent requires specific optimization mechanisms and algorithms for achieving useful down-scaling of an otherwise incomputable model (Legg, 2008). A "practically universal intelligent agent" optimizes its behaviour with respect to a given environment (or set of environments) by running iterative cycles of observation, learning, prediction, decision, action and reward (Hutter, 2012, 2013). Agent-environment interactions are modelled by formalizing both agent and environment as probabilistic functions each feeding its output to the other's input (Legg, 2008). Remarkably, general agents, built using this formalism and these techniques, are able to learn different (albeit rather simple as of now) environments without any context-related adjustments.

2.1.5 Freedom and constraint

In the context of the quest for creating synthetic intelligence, concepts of general artificial intelligence, narrow artificial intelligence, global brain and universal intelligence represent complementary research perspectives rather than competing theories. These perspectives more often interact by enriching rather than denying each other's results or theoretical approaches. Even if one of them (and "narrow" AI currently seems to be in the lead) provides an essential breakthrough for the AI research programme, it is never isolated from important influences from and repercussions to other perspectives.

One of the ways to see how the AI research perspectives interact is to identify conceptual axes along which they take distinct approaches. The following axes explain well differences and similarities:

- *Environmental interaction*: how much the behaviour of a theoretical AI agent and its implementation allows for a change, depending on the influence of an environment; a usually overlooked aspect of this axis is the degree of agent's influence upon environment.
- *Goal directedness*: how much the agent's behaviour is guided or can be explained by *a-priori* defined goals and values and how much these goals can change or adjust to circumstances.
- *Efficiency*: how important efficiency and optimal behaviour considerations are for a given perspective.

Obviously, these axes are not orthogonal – efficiency can be considered only with respect to a goal of behaviour, while development of goals and values is often related to the degree of environmental interaction. Furthermore, they are not exhaustive – i.e. there are and always will be many more axes according to which intelligent behaviour can be discriminated and analysed. Yet we can distinguish one overarching principle which grasps well the positioning of any perspective on any conceptual axis depending on how much constraint is imposed upon an agent. The principle is built upon the observation that each axis is as useful as the degree to which it enables us to grasp how much AI architecture allows for its own development in terms of interactions with environment, goal directness, efficiency and resource utilization. This principle is henceforth called **freedom and constraint**.

We can informally map all AI research perspectives in the continuum between complete freedom and ultimate constraint. The quest for synthetic intelligence can then be understood as a search for a balance between freedom and constraint in terms of specific manifestations of intelligent behaviour (agents, organisms, AI architectures, etc.) with respect to specific contexts (goals, environments, etc.). Likewise, natural evolution "searches" for the same balance in each and every species and organism.

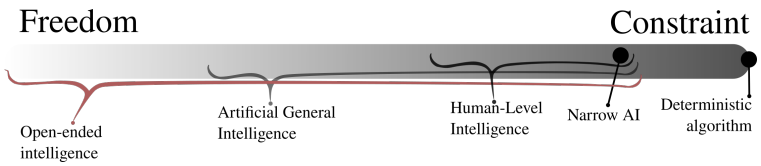


FIGURE 2.1: Research perspectives as approximately positioned on the freedom and constraint axis.

2.1.6 Open-ended intelligence

Open-ended intelligence is a novel theoretical approach to general intelligence proposed by Weinbaum and Veitas (2017b) and is:

> a process where a distributed population of interacting heterogeneous agents achieves progressively higher levels of coordination. In coordination, here we mean the local resolution of disparities by means of reciprocal determination that brings forth new individuals in the form of integrated groups of agents (assemblages) that exchange meaningful information and spontaneously differentiate (dynamically and structurally) from their surrounding milieu (ibid., p. 14).

Open-ended intelligence is a philosophical concept which first, *allows for the maximum freedom* (see Figure 2.1) and second, defines intelligence not in terms of its specific manifestations or features, but in terms of a non-linear process of bringing about *precarious balances between the freedom and constraint* invited and supported by specific contexts.

In contrast to open-ended intelligence, all of the aforementioned types of intelligence are examples of *goal-oriented intelligence*, which is characterized by (1) a more or less sharp agent-environment distinction where environment is independent of the agent's behaviour and is objectively knowable; (2) agents having *a priori* given goals while interacting in a knowable environment and (3) reward-driven behaviour with respect to goals.

The approach to intelligence as a goal-directed behaviour is well established and a prevalent mode of thinking – not surprisingly so, given its practical value in many domains, including psychology, robotics and AI research. James (1890), in his seminal study of the human mind, already chose to follow the principle that "[t]he pursuance of future ends and the choice of means for their attainment are the mark and criterion of the presence of mentality in a phenomenon" (ibid., p. 8).

While goal-oriented intelligence is the measure of an agent's competence to match actions to observations such that it will achieve optimal results in a variety of environments, open-ended intelligence is the *process of emergence of intelligence* itself, including goal-directed intelligence and its manifestations. Open-ended intelligence therefore considers maximally fluid environmental interaction by encompassing processes of agent-environment differentiation and formation of the agent's identity in the first place. Moreover, open-ended intelligence includes the processes of goal and value formation as well as determination of problematic situations which lead to goal formation. Finally, agents do not have *a priori* goals or values and interact with other similar agents in the environment shaped by the interaction itself.

The goal of this chapter is to introduce and briefly describe philosophical and theoretical concepts which are essential for conceiving actual mechanisms of the *process of becoming intelligent*. Conceiving, engineering, designing and building of intelligent machines based on the concept of open-ended intelligence is the direction of this work. The metaphysical framework of open-ended intelligence is developed in depth and breadth by Weinbaum (2018). If we ask ourselves, which domain of science and philosophy, which process observable in nature and theory most closely embraces the notion of maximum freedom, the answer is quite obvious: *evolution*. It is the best known exemplar of the open-ended intelligence process. Other domains closely related to the concept are complex adaptive systems, complexity science and network science. Therefore, we next attend to these and related domains from the

perspective of open-ended intelligence.

2.2 Evolution of body, brain and mind

Biological intelligence is a property of living organisms and has evolved with them through the evolutionary process of phylogenetic development. Importantly, living organisms are those which extend the phylogenetic development of species into the onto-genetic and cognitive development of an individual. For example, humans are considered intelligent as a species, which is the result of phylogenetic development, yet with respect to a concrete individual – an embryo or a new born baby – it is only a potentiality which has to be realized via long dependency on caretakers and interaction with the environment. Brain imaging studies have demonstrated not only that the human brain undergoes significant neurophysiological development during the first years of life, but also that it continues well beyond infancy and up to 25 years of age (Lebel et al., 2008, p. 9-10).

The perceived borderline between phylogenetic and onto-genetic is manifested by the notorious “nurture versus nature” debate of whether genes or environment cause variation in human traits. From the perspective of freedom and constraint (Section 2.1.5) phylogenetic, onto-genetic and subsequently cognitive development are phases of the same process of **progressive determination** of constraints within the initially unbounded “space of possibilities”. Progressive determination is the very central concept for this thesis which will be explained in detail later in Section 3.2.4 and referenced many times throughout the work. In a nutshell, it is a mechanism explaining how constraints that guide development of a system progressively emerge within the very same process of development – creating the "space of possibilities" of further development[2]. It is somewhat tempting to position phylogenetic development of species, onto-genetic development of an individual brain and body, and cognitive development of mind and its intellectual faculties into sequential stages. While such categorization makes sense for descriptive purposes, our aim here is the formulation of a conceptual framework of progressive determination – not for the purposes of description of how the known forms of intelligence have evolved (e.g. humans), but rather for envisioning its open-ended possibilities for evolving unknown forms of intelligence (e.g. artificial general intelligence).

Progressive determination devises a mechanism of formation of individuals which is, first and foremost, an evolutionary process. In the popular discourse, evolutionary theory is most often associated with the somewhat simplified representation of Darwin’s original work in terms of variation, selection and propagation of organisms. Yet evolution and its theory is so fascinatingly rich, multifaceted and growing from myriad internal discussions and theoretical perspectives, all having their share of empirical examples in the pot of life, that phrasing something as an evolutionary process merely means that "gods were not involved". Eldredge (2016) formulated

[2]The often used image of possibility or solution "space" is a powerful metaphor for describing the processes about which we are talking. At the same time it is somewhat misleading – a possibility space is often defined as an already existing structure which has to be *searched* or *traversed* by agent(s). Importantly however, from the perspective of open-ended intelligence, the space is not defined *a priori* - it is *created by the agent(s) in the very process of traversing it*. From a philosophical point, this is a deep distinction, worth mentioning and keeping in mind while conceiving open-ended intelligence process.

well how evolutionary theory is "[t]he elaboration of causal mechanisms underlying a process of ancestry and descent that interlinks all organisms from the inception of life to the present". While fitting into evolutionary framework, the concept of progressive determination is of course influenced by, builds on and extends selected aspects of its body. We will therefore discuss here these aspects which help to consolidate and understand the concept of progressive determination.

2.2.1 Evolution beyond biology

A rather old idea that evolution can explain the increase of complexity in domains other than biology was revived by Campbell (1997), who formulated the universal selection theory and applied it to explain the evolution of knowledge. The universal selection theory is in the spirit of Darwin's framework and states that for any fit between system and environment, three processes should take place (Heyes and Hull, 2001): (1) blind variation[3] (2) selective retention of variation and (3) preservation or propagation of variations.

Evolutionary epistemology is the term introduced by Campbell (1974) and in simple terms signifies an approach that considers knowledge creation as an evolutionary process (Gontier, 2006). The key concept in the evolutionary epistemology research programme is the one of *vicarious selectors*. It is based on the idea that nervous and cognitive systems of organisms internalize complex patterns of environment in a form of knowledge and memory. This knowledge is used for anticipating or "imagining" environmental responses and selecting those which have a higher probability of success if actually acted out – hence "vicarious". Arguably then, nervous systems have evolved precisely in order to hold the vicarious selectors in memory and map and recognize progressively more complex patterns of the organism's interaction with the physical world – which pretty much fits the definition of knowledge. Most importantly, the concept of vicarious selectors implies that variation and selection happen at more than one level – most probably in the cascade of levels where variation at each level is selected by the more or less "vicarious" selectors on the upper level. Such cascades of levels of variation and selection have acquired the name of *nested hierarchies of vicarious selectors* (Heylighen, 1995).

Evolutionary epistemology allows us to consider cognitive, cultural and social aspects of life as results of the same process that powers biological evolution and the overall increase in complexity. Remarkably, it implies that *all that evolution does is create knowledge*, albeit in different forms. Conceptually, biological organisms – bodies of plants, animals or any imaginable creatures of the world – are a form of memory, of how to live, grow, interact, procreate and die within the given environment and with other fellow organisms. The fact that part of this knowledge creation process gets extended from physical and biological into neural and cognitive, allows the application of the same principles for understanding them, with, of course, proper appreciation of the multifaceted nature of the evolutionary process itself. A particularly illustrative example of the generality of these principles is their application in computer science, AI and robotics for developing systems which create local knowledge about certain physical properties of their environment (see Box 2.1).

[3]Note, that "blind" does not mean "random" – the variation can very well be biased by, for example, developmental constraints. Yet it is always blind with respect to the selective pressures in a sense that there is always considerable uncertainty on the level of variation about which variations will be selected.

Box 2.1: Evolutionary computing and particle filters

The principle of blind-variation-and-selective-retention has been successfully reflected and utilized in the field of computer science and AI in the form of *evolutionary computation*. It is a family of algorithms based on generating large sets of candidate solutions which are then selected according to a predefined fitness criterion or function. Complete algorithms often involve many iterations where subsequent generations of candidate solutions are generated by mutating the most successful solutions in a previous generation – resulting in the evolution of the overall solution towards a target. Many variants of evolutionary computation exist, tailored for different applications and contexts. These algorithms have proved to be very successful in application domains related to high degrees of uncertainty – namely, robotics and AI, which deal with the messy physical and social world.

Particle filter is an illustrative example of how simple principles of universal selection theory are used in practice for solving contemporary robot localization problems (Thrun, Fox, and Burgard, 2005). Considering that a robot has a certain number of sensors capable of sensing its environment (cameras, radars, ultrasonic sensors, etc.) and a reference map (e.g. the plan of a building), the particle filter method works in steps resembling those of variation, selection and propagation:

i. A large number of hypotheses about robot location is generated randomly (variation step);

ii. Each hypothesis is compared to the actual readings from robot sensors and the probability of correctness (fitness) is calculated (selection step);

iii. The new set of hypotheses is generated by probabilistically choosing the ones from the old set depending on their fitness criteria – the successful hypotheses have more chance of being selected (propagation step).

iv. Steps 2 and 3 are iterated until the acceptable confidence levels for robot location are achieved.

Note, that localization is a hard problem and sometimes considered the most fundamental problem in providing autonomous capabilities for robots (Fox et al., 2001) and, hence, their intelligence. Particle filters appeared to be an extremely successful, efficient, fast and simple method to deal with uncertainties related to errors in sensor readings, ambiguous maps, movement and other real world problems. Conceptually, a robot with a particle filter can be seen as roughly implementing the mechanism of a "nested hierarchy of vicarious selectors" (see Section 2.2.1) which draws a clear parallel between creation of knowledge in natural and artificial worlds.

2.2.2 Units, levels of selection and interactions between them

In evolutionary theory, a biological entity which is the subject of natural selection is called a *unit* or *level of selection*. A heated debate persists among evolutionary scientists about what the “true” unit of selection is (Lloyd, 2012) – arguments and examples exist for treating gene, cell, individual organism, behavioural pattern, group,

species or even higher levels as such units. My position follows that of Lewontin (1970) in that "[t]he generality of the principles of natural selection means that any entities in nature that have variation, reproduction, and heritability may evolve. [... These] principles can be applied equally to genes, organisms, populations, species, and at opposite ends of the scale, prebiotic molecules and ecosystems". This does not mean that units and levels of selection are not important – rather that evolution operates on different units and levels simultaneously, albeit arguably with different levels of impact on the overall dynamics. Furthermore, the question and mechanism of interaction among different levels is of major significance.

There are a number of scientific perspectives which try to develop concepts in order to understand relations and interactions among different observed levels of organization in complex systems. Tëmkin and Eldredge (2015) explore the role of networks and hierarchies in biological evolution. Applying the principles of complex systems and the role of nested hierarchies in understanding their underlying dynamics, they endorse the hierarchical perspective to evolutionary theory. The argument is that much of the complexity of biological evolutionary phenomena stems from the synergetic effect of idiosyncratic processes at different organization levels and dynamics of inter-level interactions (Figure 2.2). Stable patterns observable in living systems are due to the "nested hierarchical architecture of the nature's economy" – as nested networks are very robust to external perturbations.

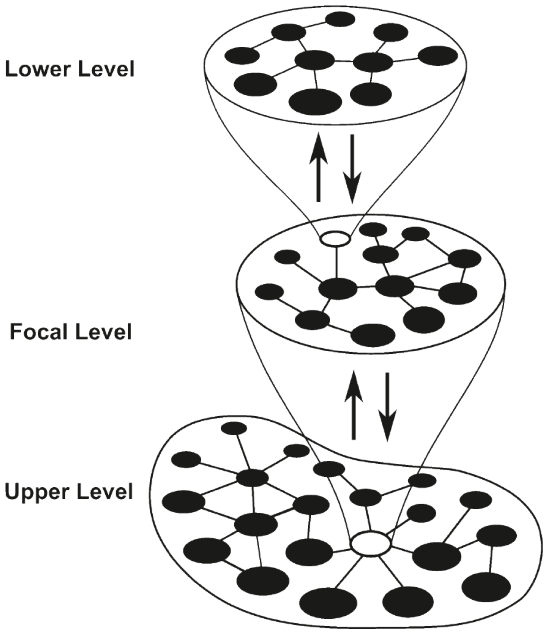


FIGURE 2.2: A diagram of global dynamics in a hierarchy, adapted from Tëmkin and Eldredge (2015). Intra-level direct interactions are shown as solid links connecting individual entities (circles) within networks at all the levels; inter-level indirect interactions representing upward and downward causation are shown as up and down arrows, respectively.

The central to complex systems science concept of *emergence* – a phenomenon of appearance of larger entities with new qualities from the interaction of smaller entities – cannot even start to be formulated without postulating distinct and hierarchically ordered levels of organization. Therefore, the model of hierarchical evolution can be, and has been, usefully applied for understanding the broad domain of

complex adaptive systems, including knowledge systems and scientific evolutions, social systems and cultural development, brain, mind and cognitive development.

In general, systems are described by combining two perspectives: *structural*, i.e. how a system "looks" in terms of wholes and parts; and *functional*, i.e. how a system "works". Not surprisingly, the hierarchical model can be applied to both perspectives in terms of structural hierarchies and functional hierarchies. Structural hierarchy is a set of hierarchical relations which describes how parts relate to the whole. Functional (or organizational) hierarchy describes a situation when one part or level of a system controls and directs the behaviour of another part or level. While both structural and functional hierarchies are observed in most complex systems, structural hierarchies are more typical for describing the nested structures of the physical world (nucleus, atom, molecule, crystal, rock, planet, solar system, galaxy and super-galaxy). Functional hierarchies, on the other hand, characterize better the world of life, mind, culture and animal or human society (Heylighen, 1995).

Structural hierarchies

The seminal article of Simon (1962) laid the ground for looking at complex systems via the perspective of *structural hierarchies*. Complex systems in the most general sense are understood as those which are made up of a large number of parts that interact in a non-simple way. In such systems, the "whole is more than the sum of the parts, not in an ultimate, metaphysical sense, but in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole" (ibid., p. 2). The analytic and descriptive power of this perspective is based on the observation that complex systems are more stable and, actually, have a higher probability of evolving their complexity when they exhibit some form of hierarchical modularity. That is, the time required for the evolution of complex forms depends on numbers, distribution and interaction of potential intermediate stable forms. Arguably, structural hierarchies themselves have evolved as an effective method for developing complex forms and, therefore, are a basic *architecture of complexity*. The argument for a hierarchical architecture of complexity is supported by probabilistic considerations and empirical observations.

Probabilistic considerations: evolution of complex systems.

Simon (ibid.) provides a simple example and simple calculations to show that a complex system has a higher probability of evolving if and when it is "organized" hierarchically from stable modules. Suppose there are two watchmakers who both are in the business of producing the same type of watch consisting of 1000 interacting elementary parts. The first watchmaker, named Tempus, has organized the process so that all 1000 parts have to be assembled at once. If the process gets interrupted by a phone call, visitor, or a small inaccuracy of assembling, an already assembled part disintegrates. The second watchmaker, named Hora, has designed a process in such a way that she can assemble 10 elementary parts into sub-assemblies, then assemble those into larger sub-assemblies of 100 elementary parts, and finally assemble the whole watch of 1000 parts. The sub-assemblies at all intermediate levels are stable in the sense that they can be securely stored. Therefore Tempus has to start the whole assemblage process from the start each time it has been interrupted,

while Hora can use previously completed sub-assemblies and repeat only the relatively small portion of work of interrupted sub-assembly. It turns out, that given these circumstances, Hora can expect to spend on average three orders of magnitude less time for constructing one watch as compared to Tempus; actually Tempus has an exceedingly small possibility of ever completing even one watch.

The mathematical formula for calculating the expected time for completing a task subdivided into subtasks is given by (Growney, 1982):

$$T = \frac{s}{p} \times [\frac{1}{(1-p)^U} - 1] \tag{2.1}$$

Where T is the expected time for completing an assemblage of a system; s – a number of sub-assembly steps, U – a number of elementary units in a sub-assembly; and p – the probability that the process will be interrupted during adding an elementary unit to a system. Intuitively, the formula says that the efficiency of a hierarchical assembly is a trade-off of probability of errors, cost of errors, the hierarchical structure of the system (in this case the number of assembly steps) and the cost of a single assembly step:

- The larger the probability and cost of an error, the more modules are needed for the same level of efficiency;
- Yet the higher the cost of an assembly step, the greater the increase in the number of modules and hierarchical levels reduces efficiency.

Even if watches were to be assembled by randomly fitting pieces together, systems having hierarchical subsystems are more stable and have more chance of occurring; therefore many natural and living systems are hierarchical. However, the metaphor assumes an *a priori* goal of the assemblage process. Yet unlike the watchmakers, evolution does not have any plan, except endless experimentation with functions *and* structures – as if watchmakers were allowed to come up with any way or mechanism that can measure time. Even then, the necessity of measuring time has to emerge from other structures and functions (e.g. agricultural, industrial society, etc.) through their evolution. We therefore can ask ourselves: what additional options and complexity is implied by the perspective to evolution as an open-ended process without goals?

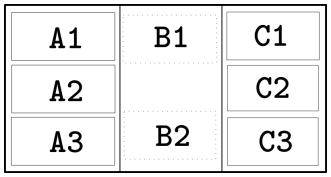
Empirical observations: "nearly-decomposable" systems

Simon (1962) also proposed the concept of the *nearly decomposable system* by observing that, in hierarchical systems, two different types of interaction of elements can be distinguished: *among* subsystems and *within* subsystems. If the magnitude of all or most important interactions among primary elements of a system are measured and recorded, it is possible to construct a matrix of interactions which would have the property of being nearly decomposable *if the system is hierarchically structured*. Informally, a decomposable matrix is a matrix containing elements that could be assembled into clusters having interactions within elements of their own cluster, but no interactions among elements of different clusters. A fully decomposable matrix would therefore represent a collection of independent subsystems which do not interact with each other. Another extreme is a system where all primary elements are interacting with the same intensity, i.e. the matrix of their interactions is not only

fully non-decomposable, but also all elements are of similar size. Such system is *flat* – i.e. it cannot be described by a structural hierarchy.

At least some of the systems can be approximated as nearly decomposable systems, which implies that the behaviour of each of their component subsystems (assemblages of elements) is approximately independent of the behaviour of other components (Simon, 1962). Near-decomposability has been observed in the broad variety of systems ranging from "natural" (produced by evolution) to "artificial" (produced by the human activity), and the ones in between. The concept has been studied in terms of economic and social structures, genetic and developmental biological models, physical and computing systems, and brains. Particularly interesting is the study of complex computing systems as nearly completely decomposable systems (Courtois and Ashenhurst, 1977).

A simple example suits well for understanding the concept of near decomposability and investigating how it relates to the architecture of complexity. Suppose that a hierarchical system under investigation is a house of a few rooms, each equipped with separate cubicles (Figure 2.3a). We can define and measure interactions between cubicles in terms of the magnitude of heat exchange between them, which could be represented in a matrix form (Figure 2.3b).



(A) House plan

	A1	A2	A3	B1	B2	C1	C2	C3
A1	--	100	--	2	--	--	--	--
A2	100	--	100	1	1	--	--	--
A3	--	100	--	--	2	--	--	--
B1	2	1	--	--	100	2	1	--
B2	--	1	2	100	--	--	1	2
C1	--	--	--	2	--	--	100	--
C2	--	--	--	1	1	100	--	100
C3	--	--	--	--	2	--	100	--

(B) Matrix representation of cubicle interactions in terms of heat exchange.

FIGURE 2.3: Illustrative example of heat exchange between cubicles and rooms in a house as a hierarchical system (adapted from Simon (1962).

The matrix in Figure 2.3 is near decomposable because it can be arranged into clusters of elements which interact much more intensely among each other than all the other elements. Obviously, in the house example such patterns are caused by the thermal insulation properties of walls between rooms and separations between cubicles. While it is somewhat a stretch to think of a house as a complex system, the same principles apply. Importantly, decomposability allows one to determine the hierarchical structure of the system without knowing its plan or blueprint – provided we can measure intensity of interactions among elements (Figure 2.4).

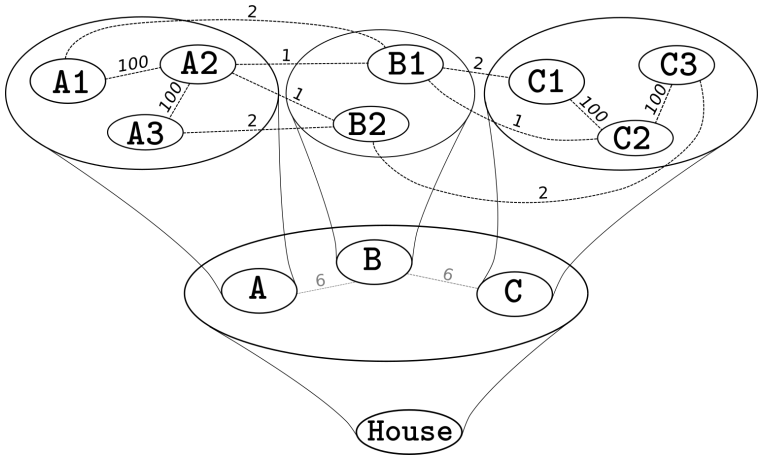


FIGURE 2.4: Hierarchical structure of heat exchanges in the house - note the resemblance to a diagram of global dynamics in a hierarchy, by Tëmkin and Eldredge (2015) (Figure 2.2).

Drawing on the above description, we can make a few interesting observations:

- First, if it were possible to calculate a single measure of "decomposability" d of a system, all systems could be positioned on a "continuum of decomposability" of a form $d \in [0, 1]$ where d approaches 0 when a system is near-decomposable (i.e. hierarchical) and 1 when it is 'flat'. Most systems cannot be said to be fully hierarchical or fully flat even if considering only one modality of interactions between elements. An example of a different modality of interaction in terms of an example with a house would be an exchange of air-flows between cubicles. A more realistic complex adaptive system may involve many modalities which could entail different hierarchical structures – see also Section 2.2.3.
- Second, the description of the hierarchical nature of a system depends on the time-scale of the measurements of interactions between elements. These two aspects are of greatest relevance to complex adaptive systems, which are highly dynamic – i.e. the magnitude of interactions between their elements is constantly changing. In such systems the measure of decomposability d may differ substantially depending on when the measurements are taken. Moreover, since the measurement of interactions necessarily involves a period of time during which the number of interactions is measured, the measure d also depends on the chosen time-scale of analysis. In other words, the short-term structure of the system may be different from the long-term structure and the former may inform little about the latter (Simon, 1962; Veitas and Weinbaum, 2017).
- Third, even if the "total decomposability" d of a system stays the same across different "snapshots" in time, the structure of clusters of interacting elements could be entirely different.
- Fourth, but not the least at all, the hierarchical structure is grounded solely in interactions among elements of the lowermost level. Heat exchanges among rooms are sums of the exchanges between cubicles of respective rooms **across their boundaries** (i.e. walls) – see Figure 2.4. It is easy to miss the importance of this observation in the example of a house with clear physical boundaries between assemblages of cubicles (i.e. rooms), where we could simply omit the level of cubicles and measure directly the heat exchange between rooms – thus

assuming the existence of real physical interactions on this level. Yet suppose a different and much more complex and adaptive system – a brain. Here, we can also distinguish three hierarchical levels: (1) individual neurons, (2) distinct brain areas (primary visual area, Wernicke's area, Broca's area, etc.) and (3) the whole brain. Yet interactions among areas cannot be explained without referring to the intensity of interactions on the neuronal level. There are no clear boundaries, "gateway" neurons or "walls" which would allow the measurement of interactions between areas directly, as in the case of the rooms of a house[4]. The importance of this nuance becomes clear when considering modelling and simulation of emergence of higher level hierarchical structures from interactions of lower elements and explaining the influence of interactions among the higher level structures on the overall system dynamics (see Chapter 3).

A conceptually similar measure to the suggested notion of decomposability d was proposed by the neuroscientist Tononi (2004) in the form of *integrated information* Φ, which will be discussed later in Section 2.3.5. In short, Φ formally defines coordinated clusters in networks of interacting agents across time and space (Weinbaum and Veitas, 2017a) and is used by Tononi (2004) in developing an integrated theory of consciousness. Intuitively, "[a] subset of elements within a system will constitute an integrated process if, on a given time scale, these elements interact much more strongly among themselves than with the rest of the system" (Edelman and Tononi, 2000). Furthermore, the first and second observations hint at the importance of the subjective perspective of the observer in seeing a system's hierarchical structure – an issue discussed further in Section 2.2.3 (in the context of model building) and Section 5.2.5 (in the context of computational models).

Functional hierarchies

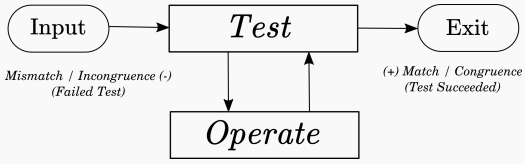
Functional hierarchies are observed when complex adaptive systems are analysed from the perspective of how their dynamics are generated – i.e. how they "work" rather than how they "look". One of the early accounts for functional hierarchies in planning and carrying out complex behaviour sequences was proposed by Miller, Galanter, and Pribram (1960) and is based on a cybernetic perspective. First, it describes the "fundamental building blocks of the nervous system" – negative feedback loops – in terms of *test-operate-test-exit (T.O.T.E) units* (see Box 2.2). A T.O.T.E unit realizes the cybernetic principle of achieving a successful goal-directed action by integrating feedback from "outside the unit" into whatever mechanism carries out the action "inside the unit". The mechanism of complex behaviour is explained in terms of nested hierarchies of T.O.T.E units, where each operational component of a unit is itself a full T.O.T.E unit which realizes the negative feedback principle at a lower level (see Figure 2.5b in Box 2.2). Similarly, the perceptual control theory developed by Powers (1973) describes nested hierarchies of negative feedback loops regulating the matching of an organism's perceptions with environmental situations. Perceptual control theory differs from the control theory in engineering as well as simple treatment of negative feedback in that it emphasizes *complex interactions between hierarchical levels*: organisms act on their surroundings and environment so as to control the effects the environment is having on them. Another account for

[4]Yet when several brains communicate between each other we again see 'real' boundaries separating one human from another.

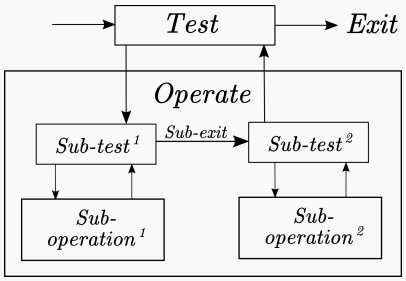
functional hierarchies inspired by the theory of natural evolution was proposed by Campbell (1974, 1997) in terms of "nested hierarchies of vicarious selectors" (as explained in the Section 2.2.1).

Box 2.2: Nested T.O.T.E units – cybernetic description of functional hierarchies

The concept of T.O.T.E units as models of negative feedback and their nested hierarchies (Miller, Galanter, and Pribram, 1960) is based on an implicit presumption that actions, behaviour and underlying mental strategies are closely related to the goal-oriented feedback loops. It maintains that actions of an organism (including mental actions) revolve around having a fixed goal and a variable means to achieve that goal. A simple T.O.T.E scheme indicates that a given behaviour is regulated by setting a *goal*, developing a *test* for figuring when the goal is achieved and then performing *operations* with the available means in order to change either the *state of the system* or the *state of environment* (Figure 2.5a). When test criteria are finally satisfied, the process exits.



(A) Basic T.O.T.E scheme. A cybernetic system accepts *input* from environment and produces *output* to the environment. It has a fixed goal (internally represented by a *test*) of changing inputs to certain outputs and variable means to perform such *operation*. The system selects and tunes the operation for achieving its goals via an "operate ⇔ test" loop by performing the operation internally and checking results against test criteria. The loop is *exited* when test criteria are satisfied – i.e. the output matches the goal of the system.



(B) Nested hierarchy of basic structures.

FIGURE 2.5: Graphical description of the test-operate-test-exit concept, adapted from Dilts and Delozier (2000, p. 1434).

Taking into consideration that the output ("exit") of a simple T.O.T.E unit can be the input to the higher level T.O.T.E unit, the general model of a functional hierarchy composed of T.O.T.E units and corresponding subgoals can be established (Figure 2.5b). The T.O.T.E model, despite its apparent simplicity, lies quietly at the roots of cognitive psychology and cognitive science and, remarkably, the symbolic perspective to intelligence and artificial intelligence.

A perspective which brings about the concept of functional hierarchies goes together with evolutionary thinking, which does not allow one to lose sight of how these hierarchies emerge in the first place. The general theory which deals explicitly with the emergence of control hierarchies in the process of evolution is the meta-system transition theory (Joslyn, Heylighen, and Turchin, 1992). A meta-system transition – the term coined by Turchin (1977) – is an evolutionary event which brings about a higher level of organization in a complex adaptive system. From the structural point of view, the new level can be said to integrate two or more subsystems at a lower level and in this sense is a *metalevel* with relation to them. From the functional point of view, the higher level is said to *control* the activity (variation)

of the subsystems at the lower level. Described in this way, a metalevel brings about qualitatively new behaviours which cannot be described in terms of characteristics of lower levels alone. Functional hierarchies emerge as a result of variation, replication and selection and can be described as nested levels of increasing control of processes happening within the system.

Box 2.3: Meta-system Transition Theory

A metasystem transition is a process in the course of which an initial system S gets replicated (with possible variations) into systems $S_1, S_2, ..., S_n$. The transition is guided by and simultaneously brings about a mechanism C which controls the behaviour and production of the $S_1, S_2, ..., S_n$ subsystems (Figure 2.6).

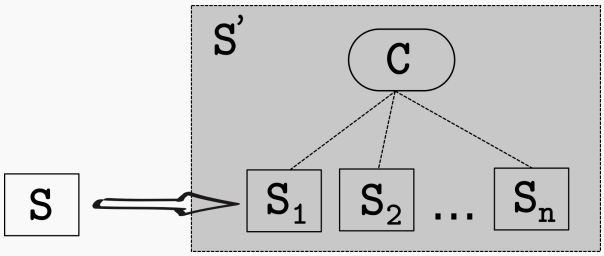


FIGURE 2.6: The metasystem transition (Turchin and Joslyn, 1993; Turchin, 1977)

From the functional perspective each control level is associated with a certain activity or aspect of a system. A metasystem transition creates a new type of activity A' by controlling the activity A of "lower" levels:

$$control\ of A = A' \tag{2.2}$$

Turchin (1977) describes human evolution as a series of metasystem transitions:

- control of position = movement;
- control of movement = irritability (simple reflex);
- control of irritability = (complex) reflex;
- control of reflex = associating (conditional reflex);
- control of associating = (human) thinking;
- control of thinking = culture.

The history of life and the universe can be conceptualized as such a sequence of meta-system transitions which lead to ever more complex, adaptive and intelligent systems spanning physical, biological, social and cultural domains. A meta-system transition can therefore be seen as a *quantum of evolution* (Heylighen and Joslyn, 1995).

Metasystem transition is first and foremost *a process*, which is rather difficult to grasp by the descriptions of its "initial" and "final" products alone. Even if it is described as such, it can only be done retrospectively. We only see control structures and functional hierarchies *after* they emerge and consolidate, yet we often miss

vocabulary and conceptual tools for grasping the intermediate states and circumstances that guide the consolidation and dissolution of such structures in complex adaptive systems.

Flat systems

Depending on how many hierarchical levels are observed (structural or functional) in a given system, it can be described as having a "flat" or "deep" hierarchical structure. For example, the house of Figure 2.3 has two lower scale hierarchical levels – rooms and cubicles. If, however, there were no rooms and the whole space was divided into cubicles, it would have only one lower hierarchical level and therefore would be completely "flat". Simon (1962) refers to hierarchical systems as "flat" at a given level if the number of non-differentiated components at that level is large (i.e. if it has a *wide span*). A completely "flat" system from the hierarchical perspective would be one where an observer was not able to identify any structure at all. As much as we tend to understand systems in terms of their hierarchical structures, there are many systems that do not have one – examples of natural systems which can be regarded as such are gas, crystals, diamonds (ibid.) or a primordial soup of elements at the dawn of life. "Flat" systems are also non-decomposable (see page 21). This means that there is no way to determine their "actual" structural or functional hierarchy because there is none in the first place. But does that mean that there is no way for an observer to make sense of a "flat" system?

Any observable system, "flat" or "deep", is also a subsystem of a larger milieu; therefore its detection is a matter of the degree of decomposability of that milieu. Even considering an example of a house with concrete walls, there are still heat exchanges and air flows which permeate these boundaries – i.e. system-environment interaction. There are systems, however, where boundaries are much more fuzzy and identification of the system itself is not straightforward – just as the decomposability of a complex adaptive system into hierarchical levels is not straightforward.

2.2.3 Hierarchies are not enough

First, let us see why the issue of hierarchies is considered important enough for such rigorous attention. It is because hierarchy in the broadest sense is brought about by an ability to distinguish and categorize – events, objects, processes, perceptions of the world – by an observer. In this sense hierarchy represents *order* and the absence of hierarchy is equivalent to *disorder*. In this sense creating order from disorder, which is a process of becoming intelligent, can be said to be equivalent (or at least closely related) to the process of seeing (or creating) hierarchies where there were none before.

The importance of a subjective perspective

In order to answer the question of what it means to make sense of a "flat" system, it is useful to think about what it means to understand (a system). Obviously, understanding (or *making sense*) always involves two parties – a system or environment which is being made sense of and an observer or an agent which is the one that makes sense. Classically, systems are considered as well defined and static structures

which are objectively given, but complex adaptive systems such as brains, markets and societies tend to be fuzzy, variable and to a certain degree "subjective" (Heylighen, 2011, p. 2). I would like to propose a working definition of subjectivity in this context: *a description of a system is "subjective" when it is decomposable in hierarchical terms only by using extrinsic discriminating criteria (i.e given from outside the system); different discriminating criteria result in dissimilar decompositions*.

The discussion of this definition and its rationale is provided in Chapter 3 with reference to the open-ended intelligence concept. For now, let us take into account the importance that the subjective perspective plays in the process of sense making and becoming intelligent.

The power and trouble of model building

The power of the concept of hierarchy is based on the empirical observation that for many natural systems, a hierarchical description considerably increases their comprehensibility (Simon, 1962). For example, it is counter-productive to try to describe interactions between every citizen of a country with citizens of another country in order to understand international relations. Likewise, the higher behaviour of an animal or a human cannot be comprehended (or even grasped) only via deciphering and mapping interactions among all the neurons in their brains. Yet any gain in comprehensibility is **always** accompanied by a certain amount of loss of information about the system being described, even when the system is trivial. It is therefore important to remember, especially given the natural tendency of cognition to search for stable patterns and invariant representations in otherwise not necessarily ordered sensory input (Hawkins and Blakeslee, 2005), the existence of this fundamental trade-off between comprehensibility and loss of information. On a highly conceptual level the phenomenon of *understanding* is closely related to the choice of this trade-off, which is less trivial than often assumed.

We usually estimate the quality of our models by measuring their predictive power – i.e. how precise is the information a model or a representation gives about the structure and dynamics of an actual system. Consider for example an agent embodied into an environment and possessing a cognitive system. For the sake of illustration let the agent be an animal, its environment - the subspace of physical reality with which an animal interacts and the cognitive system – its nervous system. One of the major aspects that allows an animal to persist and thrive in a given environment is that its cognitive system has capacities to propose adequate actions in most of the environmental situations and contexts. Such capacities are directly related to the predictive power of the internal model of the environment. Yet a precise representation of the environment is not feasible – an animal (as any bounded system for that matter) has a limited memory and therefore the whole sensory space resulting from interaction with the environment has to be efficiently *modelled* (*compressed, represented*) in order to fit into its memory[5]. Furthermore, an *adequate* action in an environmental situation implies the correct timing of the action – very often the timing of an action is more important than the type of the action (consider the classical *fight or flight* dilemma when faced with a mortal danger). To complicate the picture a bit more, there is no general way to determine the timing requirements of

[5]Such ability can be expressed as a mechanism to come up with the representation that has reasonably small Kolmogorov complexity (Li and Vitanyi, 1997).

an action in advance – i.e. a cognitive system has to be able to estimate when to stop thinking and start acting "in real-time", as the situation evolves.

Here we will concentrate on three major aspects that determine the predictive power of a model: (1) *selection for relevance* – which properties and variables out of all available options are included in the model; (2) the *degree of decomposability* of a system – how much of a system's behaviour can in principle be reasonably explained by a single model; (3) the *structural efficiency* of a model – how well a selection of properties and variables can be organized in order to achieve the same or higher level of predictive power;

These aspects determine the trade-off between the level of comprehensibility offered by a model and the degree of loss of information about a system in the process of making sense of it.

The degree of decomposability

As suggested on page 23, any system can be characterized by its degree of decomposability. The degree of decomposability of a system is its natural property – i.e. some systems are near decomposable and therefore allow for being represented in terms of structural or functional hierarchies with a relatively small amount of information loss, while others do not. Most physical systems are nearly decomposable and this fact alone has contributed immensely to the success of the scientific method. Yet there are complex systems which are not easily decomposable without significant loss of information. Non-decomposability is particularly characteristic of a class of complex which are composed of interacting heterogeneous agents, e.g. society, brain, world-wide-web, multi-agent computing systems and many more. Consider society: it is *always* the case that an individual is a member of more than one group or "category" (professional, sports, hobby, family, etc.) which cannot be easily accommodated together (Veitas and Weinbaum, 2017). These systems are of particular interest since I believe that comprehending (or *making sense of*) non-decomposable complex adaptive systems is the major ability of cognitive systems and intelligence at large. This at least means that we should not take the relatively high degree of decomposability of natural systems for granted – and that the mechanism of making sense of complex systems should be able to cope with **any** degree of decomposability.

Selection for relevance

Luckily, there is a way to make sense of a non-decomposable system – by selectively disregarding parts of information about it. Non-decomposable complex adaptive systems can be described and comprehended only by selecting some of their properties that are *significant* for the purposes of the subject which interacts with the system. The fact that decomposable or near-decomposable systems have "objective" criteria for selecting the properties and variables which allow for a very precise description of their overall behaviour should not foreshadow the need to select these variables from all available options. Such selection implies the need for criteria which can only be accounted for by considering how relevant they are for an agent making the selection decision. First of all, this introduces the necessity to consider the subjective aspect of modelling when dealing with complex systems. Second, it hints that

criteria of selection may differ across different subjects depending on their goals, which often are not uniform. Third, selection criteria of the same agent may differ in different environmental situations.

The process of selecting criteria for decomposing a given system into a hierarchical structure is closely related to what has been called *selection for relevance* in cognition (Weinbaum, 2013). Selection for relevance is an attention mechanism which accounts for a choice of attending to certain aspects of the sensory space available to a cognitive system while relatively downplaying others. Let us define the **effective sensory space** as the totality of all sensory inputs of a given intelligent animal. The **virtual sensory space** is then the totality of all possible sensory inputs that can be used for perceiving the physical reality. For example, humans heavily rely on vision for understanding the world, while bats mostly use ultrasound and echolocation – they operate in different species-level effective sensory spaces, which nevertheless are sub-spaces of the overall virtual sensory space of the physical reality. Evolution is a general mechanism of selection for relevance in terms of reducing the virtual sensory space of the physical reality to the effective sensory space of a concrete cognitive system. Likewise, at every moment and in every environmental situation, an embodied cognitive system has the task of reducing its effective sensory space into a unique combination of sensory inputs that leads to a decision and an action that make sense in that situation. Yet even a single individual cannot constantly keep full track of its own effective sensory space, at least due to the resource constraints of time and memory. A consideration that a decision for an action in a specific environmental situation is a type of knowledge, and an approach that cognition is a special case of evolutionary process of creating knowledge (see Section 2.2.1) leads to the appreciation of the importance of selection for relevance in cognition. Not surprisingly, therefore, attentional mechanisms – which are actual implementations of selection for relevance in cognitive systems – are an important aspect of research in psychology, cognitive science and AI. Even so, the selection for relevance aspect of cognition and, moreover, its evolutionary nature, are largely neglected aspects of cognitive systems research.

Structural efficiency

The structural efficiency of a model is a measure of how well given sensory knowledge about a system is decomposed in terms of maximizing comprehensibility while at the same time minimizing loss of information. It is related to the “optimization” aspect of model building – the most researched and discussed aspect of science and cognitive systems research which deals with finding the best model "fit" for a given set of information about an actual system. Note that structural efficiency here is defined not as a measure of the best model of a system, but as the *best model of available information* about a system. This is a crucial difference that takes into consideration the selection for relevance mechanism that prunes part (and maybe most) of the information about a system in order to arrive at the amount manageable by the specific embodiment of a cognitive system[6].

[6]See Section 2.3.3 on page 50 for an information-theoretical treatment of this process in terms of rate distortion theory.

Heterarchy

Heterarchy is a concept that accounts for the impossibility of describing interactions among a system's elements by positing a single system of hierarchical relations between them. A **hierarchy** is a structure of coordination among multiple agents in a system in which an agent *does not* constrain other agents, by which it is itself constrained. In contrast, a **heterarchy** is a structure of coordination among the same multiple agents in which an agent may simultaneously constrain and be constrained by other agents (Weiss, 1999). McCulloch (1945) has introduced and used the concept of heterarchy for explaining the reasonably ordered, yet non-hierarchical organization of the brain and cognitive structures. Another definition of heterarchy describes it as a relation of elements to one another when they are unranked but when they possess the potential for being ranked in a number of different ways (Crumley, 1995).

In other words, heterarchies are networks of mutual influence without subordination (Heylighen, 2002a) yet with the potential of "collapsing" into a structure of coordination that allows a system to perform computations or actions in a concrete context. This process of "collapsing" into a structure of coordination is the process of individuation (see further Section 3.2), where order emerges from disorder. We could say that three aspects play their part in this process – (1) initial conditions – i.e. the organization of "networks of influence without subordination"; (2) the emergent structure of coordination that is observable after the process; (3) the process of emergence itself. One of the most interesting findings of complex systems research is that self-organizing systems are able to perform the most sophisticated computations when operating at the boundary between randomness – i.e. first aspect – and order – i.e. second aspect (Crumley, 1995). Actually, the computation itself can be said to emerge from this boundary – hence the term *emergent computation* (Minati, Pessa, and Abram, 2009). While the focus of this work is conceiving and implementing a computational framework that accounts for all three aspects, we consider the process of emergence – i.e. the third aspect – as the most important and most neglected – perhaps due to the difficulty of approaching it. Furthermore, the subjective aspect of selection for relevance (see page 29) is instrumental for conceiving such a framework.

The fundamental trade-offs

Based on the above I suggest that a major aspect of intelligence relates to dealing with a fundamental trade-off between comprehensibility of a system's environment and the information loss about its "true" nature at each moment of comprehension. We can relate this to another fundamental trade-off, which is the primary problem solved by the whole field of computer science: the trade-off between memory and time of computation. The relating link is the concept of *bounded rationality*, designating the rational choice of an agent operating in an environment while taking into account its own cognitive limits in terms of knowledge and computational capacities (Simon, 1982). Computational capacity (equivalent to the term *computational complexity* in this context) is defined by the time and memory (space) required to perform a given computation. If we look to the act of comprehension as a computational task, the computational complexity of this task is on the one hand determined by the comprehensibility of the environment and information loss about its "true" nature (the first trade-off). On the other hand, the computational capacity –

the amount of computational resources, needed for carrying out the task – is determined by the combination of time and memory (the second trade-off). Needless to say that the computational capacity should match the computational complexity in order for the task to be completed. Memory constraint is simply a natural characteristic of a realistic cognitive system embodied in an intelligent animal or a digital computer. Time is a constraint imposed by each and every environmental situation in which a decision has to be made – e.g. making a move on a chessboard or starting to run from a tiger in a jungle. Therefore, the relation between cognitive and computational processes first of all assumes what is called *ecological view of rationality*. This view emphasizes the relation between a cognitive system and environment rather than a cognitive system and logic – since the structure of the natural environment is ecological rather than logical (see Section 2.3.2 for further explanation of the concept). In this sense, the situation determines the time constraint for a given decision and the comprehensibility of the environment determines the amount of memory (or computational/cognitive resources in general) needed for computation and permissible level of information loss about its "true" nature. Balancing these trade-offs allows a cognitive system to operate in an environment which is orders of magnitude more complex than the system.

Information compression

Comprehending a system, understanding the environment, building a model of reality, training a neural network, machine learning, statistical analysis, theory building, etc. – all these cognitive and computational activities can be seen as forms of *information compression*. Conceptually, they are implementations of a general process which takes an *external* complex, an unstructured and dynamic "corpus" of information (e.g. input to a machine learning system or an environment of a sentient being) and represents it in a much more concise form *within* a cognitive or computational system which "owns" and "runs" the process. Two different general classes of information compression processes can be distinguished: lossless and lossy compression. They are best described in a somewhat formal manner for didactic reasons in Box 2.4, but it should not prevent a view of the underlying principles in a broader sense.

Box 2.4: Lossy versus lossless information compression – definitions

Lossless information compression is the process which converts an arbitrary long string of characters (or bits) S_1 into another, strictly shorter string of characters (or bits) S_2 while preserving all information of the original string so that if the process is performed on S_2 in reverse, S_1 is fully reconstructed.

Lossy information compression is the process which also converts longer string S_1 into strictly shorter S_2, yet loses some [non consequential] information so that if the process is applied in reverse to S_2, the string S_3 is obtained, which is similar, but not equivalent to S_1.

The lossless information compression is well characterized by the measure of Kolmogorov complexity which defines the theoretical upper bound on how much a string can be compressed. More formally, the Kolmogorov complexity $K(x)$ of a finite object x is defined as the length of the shortest effective binary description of x, where $K(x)$ may be thought of as the length of the shortest computer program that

prints x and then halts (Grünwald and Vitányi, 2010). But suppose that the available storage resources do not allow the storage of even the theoretically optimally compressed string; the only option in such a case is the lossy information compression – which produces a shorter string at the cost of losing part of the original information. A lossy information compression involves a trade-off of how much information can be allowed to be lost in order to fit the result into a pragmatically viable amount of memory (or – produce a shorter string of a given length). It is trickier to define since it does not have a single implicit criterion of optimality. In the case of a lossy information compression, an optimality criterion (i.e. cost function in rate distortion theory) becomes a variable of a system which performs the compression rather than and external globally defined constraint or overarching principle. See page 34 for deeper treatment of this conceptually important, while at first sight subtle, distinction.

In summary, the cognitive operation of understanding tries to solve the dilemma about which part of the original information to keep and which part to discard (selection for relevance) together with figuring out the best representation of retained information (structural efficiency). The solution to the dilemma can be found only by considering relevant distinctions about which information is important and which is not in a given environmental situation for a given cognitive system in a given embodiment. Yet, despite an important account of the concept of lossy information for understanding the process of understanding, at least two aspects of an ecologically embedded cognitive system are not covered by them.

- First, recall that ecological (and bounded) rationality involves two constraints which both have to be taken into account: *memory* and *time*. The measure of Kolmogorov complexity and rate-distortion theory takes into account only memory (or channel capacity). However, there is a metric of complexity which considers the time aspect when accounting for lossless information compression. This metric is called *logical depth* and is best understood with relation to Kolmogorov complexity. Recall that Kolmogorov complexity $K(x)$ of an object x is the shortest program (or the shortest binary description) that can generate x. Logical depth $LD(x)$ by Bennett (1995) extends this metric by defining the *time* needed to compute x using its shortest binary description $K(x)$. Interpreted in the context of a cognitive system, the logical depth metric comes closer to considering the time needed to make a decision given the capabilities and knowledge of a cognitive system (in information theoretical terms – the time needed to decompress a compressed string to its original form).
- Another aspect is what is called *sequential dependencies* or *sequential effects* in perception and memory. In general terms this means that a local decision of a cognitive system depends not only on the current stimulus or environmental situation but also on the *history of preceding events*. Sequential effects are ubiquitous in human and animal behaviour and are experimentally very well established across a wide range of domains, including stimulus detection, perceptual identification, probability learning and decision making (Jones et al., 2013; Sims, 2016). We believe that sequential effects constitute a major aspect of ecological cognition, having broad conceptual implications on understanding the phenomenon of becoming intelligent. In the context of lossy information compression, sequential effects mean that the specific cost function for determining a trade-off between information loss and the length of the compressed information are influenced by the history of previous choices of such cost functions.

It seems obvious that the behaviour of a cognitive system depends on its history, previous experiences and encountered environmental situations, which are never the same for every cognitive system with a physical embodiment in a complex world – human or animal alike. Likewise, the principle is widely observed and applied in computer science and artificial intelligence where the performance of machine learning systems depends largely on the data which is used to train them. Yet this aspect is surprisingly often marginalized in statistical analyses as a random noise (Jones et al., 2013). The broad conceptual treatment of sequential effects is developed further with the concept of progressive determination (Section 3.2.4).

Relation to open-ended intelligence

Let us now look at how the above concepts relate to the framework of freedom and constraint in the light of how they may help to start conceiving a computationally realisable mechanism of becoming intelligent. First, goal-directed intelligence can be approached in terms of a lossy information compression with subjectively defined criteria of optimality – i.e. with the goal determined outside the framework or *a priori* given. It is not important in the sense that goal-directed intelligence considers the possibility of many goals in many environments – it still does not account for the formation of goals themselves (apart from possibly inferring a hierarchy of lower level goals from the overarching values – which are nevertheless given *a priori*). Conversely, the concept of open-ended intelligence encompasses the formation of goals and, moreover, identities of agents which have these goals in the first place as a single and inseparable process. Open-ended intelligence, of which goal-directed intelligence is a special case, can be – with huge simplification for didactic reasons – associated with framework of lossy information compression where the optimality criteria is determined *within* the framework and not *a priori*, i.e. "chosen" by the system itself. If we imagine a working model of an intelligent agent based on the concept of open-ended intelligence, the goals of such an agent would be variables of the model, acquiring their values when executing an implementation of the model but not defined exogenously.

An important note is due here. It is not possible to construct a direct implementation of open-ended intelligence. The goal of this work is nevertheless to show the possibility of "reducing" the highly abstract philosophical framework into potentially implementable ideas and thus inform the discourse about artificial general intelligence (see Section 2.1). Performing this operation will necessarily involve considering and introducing certain assumptions which are not warranted by the philosophical framework, yet are needed for closing it (as in “closure”) just enough to conceive a computational model and possibly an implementable architecture of AGI based on it. For example, open-ended intelligence does not assume agents before actual interaction happens. Furthermore, agents with their identities are assemblages of interacting heterogeneous components at a lower scale (Weinbaum, 2013; Weinbaum and Veitas, 2017b). In the philosophical framework this forms an infinite regress where the identity of every agent emerges from interaction of lower level agents. Yet, in order to conceive a computational architecture or a working model, one *has* to posit the existence of elementary components, interaction of which gives rise to further agencies and identities.

Based on the above we can start conceiving broad requirements for the cognitive

architecture based on the open-ended intelligence framework. First, such architecture should allow for the emergence of identities of higher order cognitive agents from the interaction of lower level heterogeneous components. Second, the architecture has to account for the [at least partial] ability of the emergent cognitive agents to perform the selection for relevance operation by themselves.

2.2.4 Co-evolution of structure and function

When considered in general systemic terms, evolution of body, brain and mind is a special case of evolution of assemblages of systemic components which give rise to structural and functional hierarchies. Yet, as the preceding discussion has shown, functional and structural hierarchical descriptions cannot be decoupled from each other without losing at least some properties of a system being described. Moreover, emergence and development of structural and functional hierarchies within a system implies interaction between them – evolving functions influence the developmental trajectory of structures, while evolving structures further influence the development of functions. This principle later will be conceptualised in terms of the model of progressive determination (Section 3.2.4). In this section we will discuss selected interdisciplinary domains each of which are knowingly or unknowingly associated with this principle. I am confident that this list is very incomplete.

Evolutionary development

Evolutionary development (Evo Devo) is a branch of systems thinking that holds the view that *evolutionary processes* on the one hand – which are stochastic, variety-creating, divergent and contingently adaptive – and *developmental processes* on the other – which produce convergent and systematically statistically predictable structures in a development cycle – operate together in a productive tension in order to produce change in complex adaptive systems, including, but not limited to, living systems and organisms (Smart, 2015, 2017). Generally systemic Evo Devo thinking grew out of evolutionary developmental biology (Evo-Devo) which, in its turn, had its origins in the comparative embryology of the nineteenth century (Wallace, 2002). The major proposition and argument of evolutionary developmental biology is that the development of embryos is shaped by their phylogenetic and ontogenetic developmental trajectory no less than selective pressures or inductive interactions within an embryo (Kalinka and Tomancak, 2012). A developmental trajectory itself is shaped by many interactions – between individuals of the same species, different species, species and their environment, etc. – which are very complex (Hall, 1999). In other words, every and each phase of development is influenced by developmental constraints, contingently emerging in the course of the previous phase, as well as the myriad interactions inside and outside the boundary of an embryo – a developing living and complex adaptive system.

Developmental and selective constraints

What are called *developmental constraints* represent a bias on the production of variant phenotypes or a limitation on phenotypic variability caused by the structure, character, composition, or dynamics of the developing complex adaptive system (Smith et al., 1985). Developmental constraints modify evolutionary future at every point

in the process and therefore introduce a bias on what kind of developmental constraints further arise. Approached from the perspective of progressive determination of developmental constraints in an evolutionary process the "nurture versus nature" debate loses at least some of its tension, because whatever constraints are stored within an organism are the result of the interaction between an organism and its environment at some stage of the evolutionary developmental process. This also implies that the very boundary between an organism (whether in terms of its genotype or phenotype) and environment is determined during, inside, and *because of* the process of evolutionary development rather then exogenously.

Niche construction

The crux of classical evolutionary frameworks of natural selection, such as the universal selection theory of Campbell (1997) (see Section 2.2.1) is the principle that living systems evolve via blind variation of their features and selective retention of those which lead to successful skills and behaviours in an environment. The framework is based on two strong implicit and intertwined premises: (1) that the environment is (at least largely) stable and (2) that developing organisms do not influence their environment. Yet if we realize that the environment of an evolving and developing organism is composed of other evolving and developing organisms, it becomes obvious that we are looking at interactive co-accommodation rather than one way adaptation. Adaptation is such a powerful explanatory paradigm because mutual influences among organisms and their assemblages are far from symmetric. For example, the biosphere of a certain planet called Earth can be considered a living and evolving organism, yet the time-scale of development of this biosphere is so different from the time-scale of, say, a mammalian life form, that thinking in terms of adaptation of the latter to the former is a good enough approximation of what actually is a mutual interaction[7].

The theory of *niche construction* – a term coined in the late 1980s by Odling-Smee (1988) – addresses precisely this dynamic. Niche construction is the generalized process where organisms actively modify their own and other organisms' evolutionary niches while simultaneously adapting to them. An evolutionary niche is defined as the sum total of all the natural selection pressures to which a population is exposed (Odling-Smee, Laland, and Feldman, 2013) – a concept closely associated with what was called an "effective sensory space" of an organism on page 30.

The four key tenets of niche construction theory are (Laland, Matthews, and Feldman, 2016):

i. organisms impose a systematic bias on the selection by modifying environmental states in non-random ways – thus exerting influence over their own evolution;

ii. parents influence the ecological niche of their offspring thus contributing to parent-offspring similarity; this gives rise to *ecological inheritance*, operating besides and together with genetic inheritance;

iii. acquired characters and by-products become evolutionarily significant by affecting selective environments in systematic ways, and;

[7] Yet the global climate change is the evidence of how some mammalian forms can influence the biosphere.

iv. the complementarity of organisms and their environments (traditionally described as "adaptation") can be achieved through evolution by niche construction.

An important aspect of the theory, which attracted a fair amount of criticism from standard evolutionary theorists and a subsequent debate, is that niche construction is not predictive, at least not in a strict sense – because it can lead to both long term increase or decrease of fitness (Scott-Phillips et al., 2014). Developmental processes of niche construction channel selection along particular evolutionary pathways which individuate during the process itself and, therefore, are at least partially unpredictable. Indeed, niche construction theory emphasizes that "organisms actively contribute toward both the construction and destruction of their own and each other's niches" (Odling-Smee, Laland, and Feldman, 2013) and that both these directions should be taken into account. Precisely such a dynamic, yet in much more abstract terms, is addressed within the philosophy of individuation and theory of assemblages (see Section 3.2.2).

Besides biological evolution, niche construction theory has found a fertile ground in psychology and cognitive sciences. Concepts like *cultural niche* and *cognitive niche* relate to environmental aspects of non-biological origin, which have an important influence on human evolutionary dynamics (Pinker, 2010). These aspects are clearly shaped by human behaviours, and therefore some theories in cognitive science started to focus on the active role of organisms in shaping their own cognitive niche – collectively and individually. Approaches of situated, embodied, ecological, distributed, extended and enactive cognition look beyond "what is inside a person's head" to "what a person's head is inside of" and with which it forms a larger whole (Clark and Chalmers, 1998; Di Paolo and De Jaegher, 2012; Stotz, 2010).

Developmental Systems Theory

Developmental systems theory (DST) is the general theoretical perspective on biological development, heredity and evolution – a framework for conducting scientific research and understanding the broad significance of its results. DST tries to disassemble dichotomies of "nature versus nurture", "genes versus environment", "biology versus culture" by viewing development and evolution as a "process of *construction* and *reconstruction* in which heterogeneous resources are contingently, but more or less reliably reassembled for each life cycle" (Oyama, Griffiths, and Gray, 2001). DST understands organisms as autocatalytic systems – systems which are able to self-organize and self-maintain not because they are adapted to the environment (Gontier, 2006), but due to inner mechanisms that enable them to self-maintain a metastable homeostasis within boundaries of survival – sometimes *despite* the environment.

While DST grew out of the application of general systems theory to embryology, epigenetics and developmental psychology (Griffiths and Tabery, 2013), it can be considered a broad *scientific paradigm*, also referred to as *developmental science*. As such, DST has been applied to developmental psychology, biological systems theory, econometrics, systems science, psychometrics and more. Additionally, there are strong conceptual links between DST and other theoretical models and research methodologies, such as dynamical systems, artificial neural networks, connectionism, theoretical systems modelling, simulation modelling, system dynamics, social

network analysis and agent-based modelling (Molenaar, Newell, and Lerner, 2013).

Oyama (2000), a major proponent of the DST paradigm, has systematically developed the idea that developmental information is actually produced during development – i.e. *information has an ontogeny and a developmental history* (Griffiths and Tabery, 2013). Building on this idea alone turns developmental science and DST into a scientific paradigm deeply associated with the process philosophy and philosophy of individuation. A major question of any scientific investigation, perhaps mostly palpable in the context of evolutionary theory, is *how a form arises*. Oyama (2000) distinguishes three avenues for approaching the question:

i. **"Preformationist" attitude**: there is a single causal source which defines *a priori* any form – a process that brings it into existence is merely mechanics of unfoldment of already existing information. This perspective is embraced by the view that genes or any single principle are responsible for organismic forms.

ii. **"Interactionist" attitude** – there are several causal sources which interact in a complex way in a process of unfoldment of a form, the main question being what these sources and their relative influences are. Similar to the "preformationist" attitude, information exists *a priori*, but in a distributed fashion. This perspective is embraced by epigenetics.

iii. **"Constructivist interactionism"** – there are no prior causes or sources of information preceding the developmental process which results in a form. Evolutionary development is not a process that takes information as an input from one or more sources and combines it into a form – information itself develops during the process.

Obviously, DST embraces the "constructivist interactionist" attitude:

> Developmental information [...] neither preexists its operations nor arises from random disorder. It is neither necessary, in an ultimate sense, nor a function of pure chance, though contingency and variation are crucial to its formation and its function. Information is a difference that makes a difference (G. Bateson, 1972, p. 315), and what it "does" or what it means is thus dependent on what is already in place and what alternatives are being distinguished (ibid.).

Although never mentioned by Oyama (ibid.), the "constructivist interactionism" attitude fits perfectly within Simondon's theory of individuation (see Section 3.2), which is much more general and not *per se* coupled with evolutionary biology. At the core of the theory of individuation and the entire Simondonian ontology is the philosophy of information as a new "systemic and not cybernetic" theorization (Barthélémy, 2015; Simondon, 2009).

Complexity science and self-organization of complex adaptive systems

Complexity is the paradigm within science and philosophy that studies general-systemic principles of dynamic, evolving and developing (in terms of structure and function) systems, i.e. *complex adaptive systems* (CAS). CAS are characterized by complex patterns of behaviour which emerge from interactions among a large number of component systems (agents) at different levels of organization (Ahmed, Elgazzar, and Hegazi, 2005; Chan, 2001; Gell-Mann, 1994). Outcomes of a huge number of interactions are most often unpredictable due to their non-linear character. Still,

these interactions are able to spontaneously coordinate among each other – therefore, complex adaptive systems are said to *self-organize* instead of being organized or designed.

Self-organization is the appearance of structure or pattern without an external agent imposing it (Heylighen, 2001b). Importantly, self-organization is caused by a certain amount of disorder and fluctuations in a system – as formulated by principles of "order from noise" by Heinz von Foerster and "order from fluctuations" by Ilya Prigogine (ibid.). These principles point to an important understanding that fluidity, disorder, fluctuations and uncertainty are not undesirable side effects which should be minimized, but actually are necessary for a complex adaptive system to evolve and thrive. We can visualize a process of self-organization as a series of symmetry-breaking bifurcations of a complex adaptive system operating "at the edge of order and chaos":

> One might expect that the problem of how nature generates pattern and form would be to explain how symmetry arises out of chaos and disorder. But in fact, disorder is much more symmetrical than order. If a beautiful bronze sculpture is melted down into a uniform pool of liquid metal, its form and structure are lost—but it gains a great deal of symmetry. Thus the question of the genesis of form is not how symmetry arises out of disorder, but rather how the symmetry of disorder gets broken in determinate ways to produce the characteristic asymmetries of the forms we find in nature (Brender, 2013, p. 267).

Complexity science as a whole is still mostly a collection of exemplars, methods and metaphors of modelling complex adaptive systems without a generally accepted definition (Heylighen, 2009a; Mitchell, 2006). Despite the difficulties in formalizing the notion of complexity, CAS can be characterized by somewhat more intuitive features, the major of which are:

i. Complexity must be *situated in between order and chaos*, sometimes called an "edge of chaos". Since complex systems are neither regular or predictable, nor random or chaotic, a number of theorists have proposed that the precarious balance between order and chaos is precisely what is necessary for adaptation, self-organization, and life to occur. Complex adaptive systems tend to evolve spontaneously towards this balance.

ii. Complex systems consist of many components that are connected via their dynamic interactions (not static or clearly definable links or relations). These components can be said to be partly autonomous while at the same time partly mutually dependent.

iii. Dynamic interactions among many heterogeneous components give rise to *emergence* – behaviours and properties at the scale of the whole system which cannot be described via reductionist analysis of properties and behaviours of individual components.

This intuitively defined notion of the complex adaptive system encompasses a broad array of natural and artificial systems and phenomena that can most generally be described as "living" – including organisms, societies, brains, minds, intelligent agents and their communities, languages, business organizations, nation states, ecosystems and many more. They are best approached and studied via simulation modelling techniques based on agent-based models, where many heterogeneous agents form complex networks of interactions giving rise to the emergent properties of a

whole system (Di Paolo, Rohde, and De Jaegher, 2008; Heylighen, 2009a; Komosinski and Adamatzky, 2009; Weiss, 1999). This avenue is followed throughout this thesis and detailed in Chapters 3 and 4 for the purpose of conceiving, modelling, studying, understanding and possibly implementing an (artificial) complex, living and intelligent system.

(Anti)fragility

A new and highly conductive perspective to complex adaptive systems is proposed by Taleb (2012). This perspective characterizes a system by its long- and short-term response to external perturbations. A number of common notions used to describe the vulnerability of a system or "thing" are explained in the following way using this perspective:

- **Fragile** applies to a system that disintegrates or loses its properties immediately when the energy of perturbation exceeds certain threshold. Taleb (ibid.) gives the example of a porcelain cup, which breaks easily from contact with another object or falling to the ground – i.e. exposure to a small perturbation. Yet an engineered system which can withstand most of the perturbations but breaks due to unusually high energy is also fragile, albeit with much higher fragility threshold. Fragility is therefore a dynamic property of the system which describes not only the threshold at which it breaks, but, most importantly, the dynamic response to perturbations that are smaller than the threshold. Depending on how large the threshold is, systems may be considered **vulnerable** or **resistant** (Johnson and Gheorghe, 2013).
- **Robust** or **resilient** systems are the ones which do not change when exposed to a perturbation of smaller energy than their fragility threshold.
- **Antifragile** systems and things respond to the perturbations of lower level than the fragility threshold by re-arranging their internal organisation in a manner that makes their fragility threshold *higher than before perturbation*. Therefore antifragile systems are said to "positively thrive on uncertainty". Many complex adaptive systems and all living systems are antifragile.

Apart from adding the ecological perspective to the complex adaptive systems and their relation to the world (Veitas and Weinbaum, 2015), the concept of antifragility and its measures have been mathematically defined and applied for risk management and financial institutions' stress testing designs (Taleb et al., 2012; Taleb and Douady, 2013).

Network science

Network science is a relatively new discipline which gained weight and popularity at the beginning of the 21st century, mostly due to its applicability for studying systems with millions or billions of interacting components – societies, the internet, molecular, metabolic networks and more (Barabasi, 2013). Despite the lack of a generally accepted definition of CAS and complexity science, many real world complex systems can very well be represented as networks. Recall that complex systems consist of many elements connected via dynamic relations. Likewise, a network (or a

graph) is a collection of nodes and links between them, which can be directed or undirected, weighted or unweighted, typed or untyped.

Network science and, more specifically, adaptive networks, allow the structure and dynamics of those systems to be abstracted from their immediate domain and in this way to be studied mathematically (Sayama et al., 2013). This fact alone provides a powerful avenue and a unified metaphor for modelling, simulating and understanding complex adaptive systems. Surprisingly many, if not all, real world systems can be modelled as network maps – a collection of objects (nodes, vertices) and relations between them (links, edges) (Rodriguez and Neubauer, 2010). Networks can be said to be skeletons of complex systems, which, however, have to be dressed with dynamic interactions in order to model these systems faithfully (Barabasi and Frangos, 2002). Agent-based models and multi-agent networks, broadly used for modelling and simulating complex systems, can actually be mapped with chosen precision to graph structures, where agents are represented as nodes and their interactions as links (Heckel, Kurz, and Chattoe-Brown, 2017; Ren and Cao, 2011).

While network science is a new discipline, the concept of the graph was introduced as early as the late 18$^{\text{th}}$ century. The beginning of graph theory in mathematics is considered to be even earlier and started with Leonhard Euler's paper on the Seven Bridges of Königsberg (1736). Graphs are one of the primary structures in the study of discrete mathematics and have a number of mathematically proven properties with well defined metrics. Yet the notion of the graph used in contemporary complex systems modelling and network science may considerably differ from the notion commonly used in mathematics[8]. The reason is that real world complex system networks, apart from being dynamic and adaptive, consist of many heterogenous agents that interact in different ways or may have interactions of different types. For example, a social network may consist of friends and colleagues who interact in the form of casual chats, beer evenings, or work assignments and scientific discussions. Furthermore, the same two members of a social network can engage in different types of interaction. These relationships are not grasped by the simple graph structure commonly used in discrete mathematics.

Over the two centuries of graph theory history, many flavours of graph structures were invented in order to represent different properties and problem spaces (see Rodriguez and Neubauer (2010) and Figure 2.7 for a wide, yet still incomplete list of graph types). For our purposes it is enough to distinguish three types, representing extremes of the spectrum:

(1) *Ordinary or simple graph* is a tuple $G = (V, E)$ where V is a set of homogeneous nodes and $E \subseteq V \times V$ is a set of edges that connect a pair of nodes (Kivelä et al., 2014). This structure is the most studied in graph theory and the great majority of mathematical proofs are based on it. Despite being the most accessible for mathematical representation and reasoning, the simple graph is limited in its expressiveness and less suitable for representing interesting real world systems.

(2) *Multilayer graph* is an extension to the simple graph developed in order to represent and model more complex and closer to real world systems. In a most general form a multilayer network is a quadruplet $M = (V_M, E_M, C, L)$, where:

[8] Francis Heylighen (2016), private conversation.

(a) V is a set of non-homogeneous nodes;

(b) $V_M \subseteq V \times L_1 \times ... \times L_d$ is a set of layers in which $v \in V$ is present;

(c) $E_M \subseteq V_M \times V_M$ is the set of edges containing the pairs of possible combinations of nodes and elementary layers;

(d) $\{L_a\}_{a=1}^{d}$ is the set of elementary layers defined by d aspects.

The traditional metrics of simple graphs can be generalized to work with multilayer graphs, yet the generalization process is complex and involves advanced mathematical tools, such as tensor algebra, generating functions and spectral theory (Tomasini, 2015). Additionally, these graph structures are quite cumbersome and not very intuitive.

(3) *Property graph* can be seen as a further extension of multilayer network in that it allows for an arbitrary number of edge and vertex labels (types) which can therefore be heterogeneous in meaning. It also allows for an arbitrary number of properties that can be attached to edges and vertices. Formally, a property graph is defined as $G = (V, E, \lambda, \mu)$, where:

(a) V is a set of non-homogenous nodes;

(b) $E \subseteq V \times V$ is a set of directed edges;

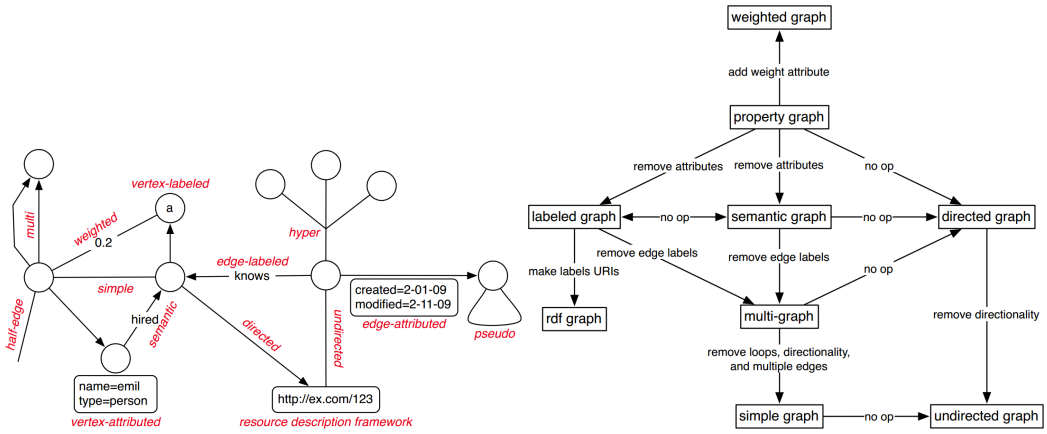
(c) edges are labelled (i.e. $\lambda : E \to \sigma$; where σ is a text label);

(d) properties (labels or types) are a map from elements (nodes and edges) to keys and values (i.e. $\mu : (VE) \times R \to S$; where (VE) are elements, R are keys and S – values).

The property graph is the most expressive graph structure and by its use all other graph types can be represented (see Figure 2.7b) [9] This is one of the reasons for our use of property graphs in the computational model of open-ended distributed computing (see Chapters 4 and 5).

Hierarchical structures of systems, discussed in Section 2.2.2, are formally *trees*, which are special forms of graphs that do not contain cycles and where every vertex has only one outgoing link. Finding a decomposition (i.e. hierarchical structure) of a nearly decomposable system amounts to finding a spanning tree on a graph of this system, which is a well defined computational operation (Ruohonen, 2013). Recall from discussion on page 21 that most of the physical systems are nearly decomposable. Complex systems on the other hand, are non-decomposable in a strict sense, yet their decomposition *can* be found if one accepts the cost of incurring an arbitrary amount of information loss. We will explore this avenue in detail in Chapters 4 and 5. For now it is important to establish the natural applicability of graph structures (concretely the property graph) for dealing with the decomposability of complex systems and therefore finding hierarchical and heterarchical structures in them which, as was proposed earlier (see page 29), is the major operation of intelligence.

[9] Including a *directed labelled hyper-graph* which is a comparably expressive graph structure, where links can connect more than two vertices. Nevertheless, it is in principle possible to represent any hyper-graph structure using property graph formalism, yet not the other way round. A labelled directed hyper-edge can be represented as a special vertex which is then connected to other vertices via simple binary edges. In property graph, every such binary edge could have a different label, yet this is not possible to represent in a hyper-graph. On the other hand, certain structures can be represented in a more compact way using hyper-graph formalism.



(A) Possible formalisms that can be combined in order to obtain different types of graph structure.

(B) Expressiveness of major graph types with weighted property graph as the most expressive formalism that can be morphed to other graph types via selecting out certain properties.

FIGURE 2.7: Description and comparison of different graph types. Adapted from Rodriguez and Neubauer (2010)

Perceptual control theory

Perceptual control theory (PCT) is a theory of behaviour which was conceived in the 1950s by Powers, Clark, and McFarland (1960) and is still in active development. It has been applied to diverse fields, including animal behaviour, neuroscience, sociology, psychology, psychotherapy, robotics, human-machine interaction and more[10]. PCT differs from other theories in its basic assumption about the nature of behaviour itself and is sometimes considered the "third grand theory of behaviour" along the stimulus-response model of behaviourism and information processing model of cognitive psychology. Yet for a long time it was overshadowed by behaviourism and cognitive psychology, to which it was opposed, and only started to re-emerge thanks to the rise of self-regulation theory (Mansell and Marken, 2015).

The fundamental idea of PCT was known to Aristotle (and probably much earlier) and well expressed by William James as: *people act so as to bring conditions they desire – to perceive their world as they wish it to be* (Taylor, 1999). Yet this idea gained the backing of technical knowledge and system theory only after the introduction of the notion of cybernetics by Wiener (1950) and subsequent developments in the field of engineered control systems. The core tenet of PCT is that *all behaviour is control of perception*. Control in the strictly engineering sense is defined as bringing a perception of some state to a desired (reference or goal) to which it is compared and maintaining it there (Taylor, 1999). PCT derives quite a few fundamental principles that make it stand out from control theory in engineering as well as other theories of behaviour. We here briefly describe a few aspects which are most relevant to the present work without trying to be exhaustive or complete in covering this 70-year-long research programme (Powers, 2016).

The core of PCT is well described in terms of a 'closed-loop' cybernetic system yet contrasting it with the traditional models of cybernetic control. The first difference lies in that while traditional models of system control assume that a system controls its *output*, PCT assumes that a system controls its *input* – which literally follows from

[10]See e.g. http://www.pctresources.com/index.html

its core tenet that behaviour is control of perception, and perception is obviously an input.

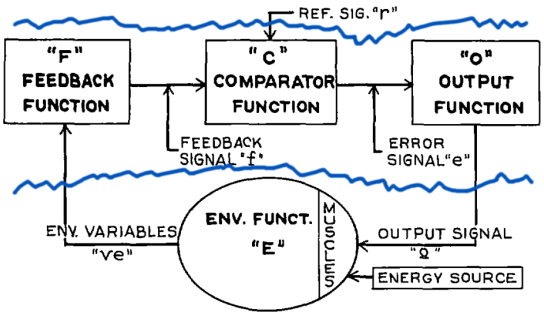


FIGURE 2.8: A general form of a feedback control system (adapted from (Powers, Clark, and McFarland, 1960, p. 76)). Blue lines indicate usually perceived environmental boundaries that separate an individual feedback loop from its environment – a relationship that is often simplified in order to make things more 'understandable'.

At first glance, Figure 2.8 looks similar to the classical depiction of a control loop, but there is a key difference. Internally (between blue lines in the figure), a feedback loop consists of three functions: (1) a *feedback function* $f = F(v_e)$ which converts the environmental variables v_e to the feedback signal f to be further considered by the system; (2) a *comparator function* $e = C(f, r)$ which compares the feedback signal f ("perception") with a reference value r ("desire") and produces an error signal e. Usually the comparator function is considered a form of subtraction between f and r, but this need not be the case in general; (3) an *output function* $o = O(e)$ which conceives some sort of external action by the system. Classically, a control loop tries to correlate the output signal with the environmental variables in order to minimize error. In PCT, the nature of correlation is modulated by the reference signal r. Furthermore, PCT greatly relaxes the "freedom" of a control loop to minimize e by enabling the adjustment of **all three functions** – i.e. being able to perceive, compare and correct via environmental loop variables that maximize its perceptual control. In this sense, the seeming "adaptation" of negative feedback systems to environmental signals is only a side effect of being persistent in this environment – the goal of such correlation is not in any way directly represented in a system. I believe that this insight carries great importance when considering *living cognitive* rather than engineered systems.

The second crucial difference is that in the PCT control model the goal specification of the system can be said to be set *in and by the system*, while in the traditional model it is set exogenously (Mansell and Marken, 2015). Formally, goal specification is the reference value of a chosen control variable – so, for example, if a system is a thermostat, its goal specification is the difference between *perceived* temperature and *desired* temperature levels. When the goal is specified exogenously, a system can only aim at its perceptual goal by somehow changing the variable itself (*actual* temperature in case of thermostat). When the goal is specified endogenously, a system can aim for the same perceptual goal also by changing the perception or desired level of a variable without trying to control the variable itself.

The above may seem to make little sense in the context of control systems which are engineered specifically with the purpose of controlling an exogenously set variable (as it is in the case of thermostat – probably nobody wants a device that decides by itself how to perceive and react to the temperature in a room). Yet in the context

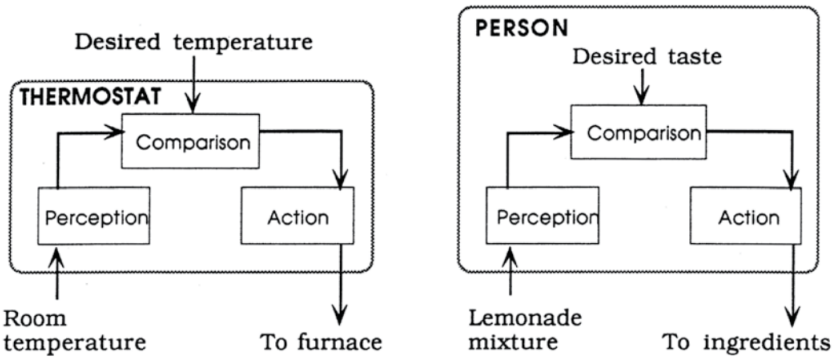


FIGURE 2.9: A functional comparison between a thermostat and a person, showing the difference between them in terms of the source of the desires that animate them. Adapted from Powers (2016, p. 284).

of an autopoietic and self-regulating system with the purpose of regulating a precarious balance between order and chaos in its own behaviour, it does not really matter how it is done as long as this balance is kept and the system manages to persist in a given environment. In this sense, PCT operationalizes into a falsifiable theory the principle that the driving force behind living cognitive systems' behaviour is to perceive the world the way they wish it to be.

The third difference in PCT from traditional theories of cybernetic control is the emphasis on interactions between multiple levels of negative feedback units ("closed-loops" of Figure 2.8) – leading to understanding that (1) these loops can still be "open" to influences from higher levels and (2) that the process driving their "individual" closure is a dynamic property of the whole system – consisting of multiple hierarchical levels of feedback control subsystems (see Figure 2.10).

Multiple levels of feedback control loops in PCT are almost equivalent to the concept of nested hierarchical levels of test-operate-test-exit loops of Miller, Galanter, and Pribram (1960), described on page 24. Yet PCT goes further than stacking feedback control units on top of each other and connecting their inputs and outputs. It attempts to describe complex relationships between orders, which are "responsible" for the individual formations of "closed-loops".

This is done by introducing an additional *recording* function R to each individual feedback subsystem, resulting in the functional description represented in Figure 2.11. In human terms, the recording function is responsible for memory and imagination, which are fundamental for the completion of the PCT model. The recording function can be approximated to something like $f_r = R(f, r)$ (using the terms introduced above), where f is a regular (current) feedback signal, f_r – a *recorded* feedback signal and r is a reference value. The recording function has the property of returning the memory of a signal experienced by the system in the past when triggered by circumstances considered external to the local system. The bottom line is that this extension (1) allows the system to have some degree of non-trivial "choice" between the current f or recorded f_r feedback signal as an input to the comparator. Also, (2) it relates levels of feedback control units by positing that reference values – signals r from higher order systems – no longer serve directly, but only through stimulating memory-trace in recording function. The comparator function is indifferent as to whether it receives an "imagined" memory or the currently perceived feedback signal. This property allows the system, in human terms, to "dream", "hallucinate",

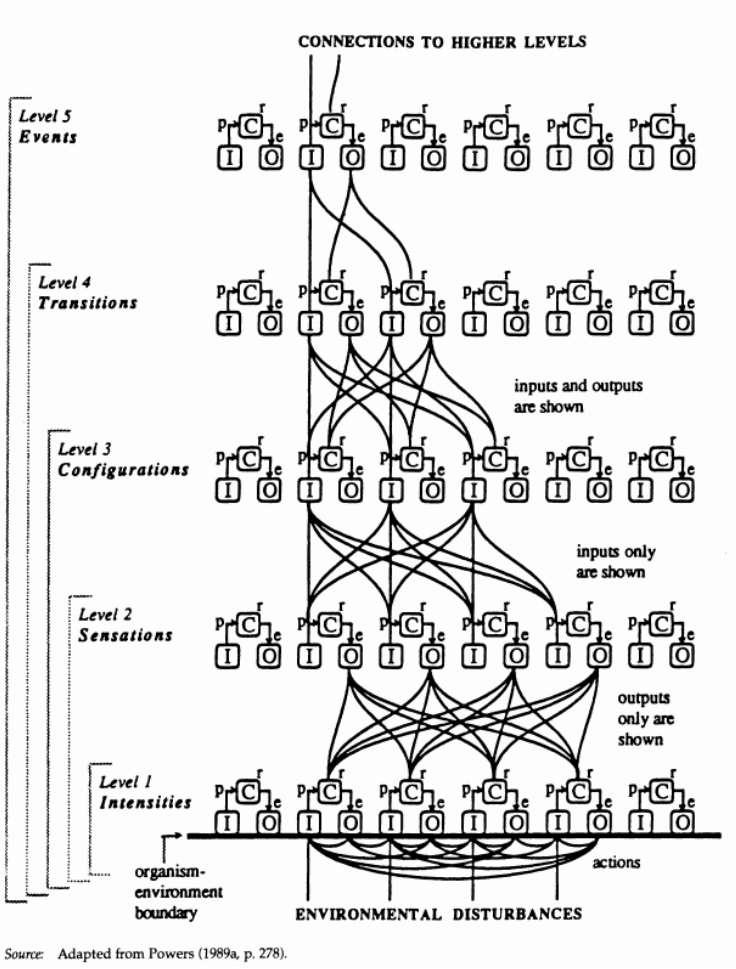


FIGURE 2.10: Interconnections in a control system hierarchy. Adapted from McClelland (1994, p. 470).

"fantasize" and to "plan" actions mentally. This kind of "choice" is closely related to what was earlier presented in more abstract terms as the *selection for relevance* aspect of model building (see page 29).

Remarkably, the epistemology of PCT, as well as the theory of autopoiesis, is constructivist: an organism's knowledge should not be seen as an objective reflection of outside reality, but as a subjective construction, intended to help find a way to reconcile the system's overall goal of maintaining its organization with the different outside perturbations that may endanger that goal (Heylighen, 2002a). The hierarchically organized system of multiple orders of feedback control sub-systems (called the negentropy system or *N-system* in PCT) is capable of learning via coming up with stable organizations of relations between orders. In terms of evolutionary theory, the learning process can be understood as a mechanism of internalizing vicarious selectors into a system (see Section 2.2.1). Yet such a system has a fluid structure which becomes a stable feedback system "not because there is anything that 'tells' the system to stop reorganizing, but because the lower-order systems and the environment are such that this particular organization produces behavior which results in a lessening of the intrinsic error, thus slowing or halting the reorganization process" (Powers, Clark, and McFarland, 1960, p. 82).

Furthermore, PCT defines a system as a set of functions interrelated in a special

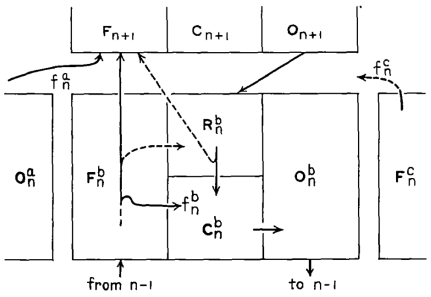


FIGURE 2.11: Relations among orders / strata of the system. Adapted from Powers, Clark, and McFarland (1960, p. 83).

way. An environment of a system can then be defined as all those functions and variables not included in a set chosen to define a system (Mansell and Marken, 2015). A system interacts with an environment via its boundary, defined as input boundary (i.e. all "inward" functions which relate environmental variables to system variables) and output boundary (i.e. all "outward" functions which relate system variables to environmental variables). Note that this definition depends on the choice of the boundary, which is at least partially observer dependent.

Last but not least, perceptual control theory considers the collective control of individual feedback subsystems to be the main mode of operation, as can be seen from Figures 2.11 and 2.10. A higher order subsystem receives environmental signals v_e from more than one lower order subsystem and a lower order subsystem receives reference signals r from more than one higher order subsystem. One can further imagine lateral relations between subsystems of the same order in the spirit of the hierarchical temporal memory / cortical learning algorithm of Hawkins and Blakeslee (2005). Thus defined, collective control further emphasizes and allows one to appreciate the fuzzy nature of system-environment boundaries, and, moreover, the boundaries among hierarchical levels within a system (conceptualized as *scales of individuation* in the Section 3.3.2). The PCT model of collective control has inspired a body of research and validation attempts in the area of social systems (Busseniers, 2018; McClelland, 2006).

Stigmergy

The concept of *stigmergy* was originally introduced in order to account for the complex coordination observable in biological insect societies (Grassé, 1959), but has since been generalized into a universal mechanism of decentralized coordination in societies of interacting individuals (Heylighen, 2015b, 2016) and even a probable counterpart of the natural selection process (Vidal and Dick, 2014) in evolution. As defined by Heylighen (2016), "stigmergy is an indirect, mediated mechanism of coordination between actions, in which the trace of an action left on a medium stimulates the performance of a subsequent action".

More concretely, stigmergy is a form of indirect coordination between independent actors via a shared medium, where some actors leave a trace that is picked up and acted upon by other actors and in such manner guides their performance. It is probably the simplest yet most effective form of coordination of complex systems – so effective that it gives rise to the phenomenon of *collective intelligence* famously

observed in large colonies of eusocial insects, where a group is observably more intelligent than any single individual in it. A key requirement for the stigmergy to occur is the presence of the *shared medium* – a regulatory structure external to the agents that promotes coordination (Heylighen, 2011, p. 33), where signals can be independently written and read by the participants of the system.

The concept of stigmergy has been applied for understanding coordination in colonies of simple organisms, molecular interactions, biological and robotic swarms, cognition and many more domains. Furthermore, stigmergy attracted the attention of computer scientists when it was noticed that, by using it, eusocial insects collectively and without prior knowledge actually solve complex computational problems, such as the travelling salesman problem, which is an *NP*-hard combinatorial optimization problem (Stützle and Dorigo, 1999). This phenomenon has been called by computer scientists *stigmergic optimization* and *stigmergic computation* and is extensively researched (Abraham, Vitorino, and Grosnan, 2006; Pintea, 2014) from the computational point of view. This perspective is most interesting in the context of this work as it is instrumental for the computational framework of open-ended intelligence developed in Chapters 4 and 5. For the conceptual treatment of stigmergy as well as its applications in other domains see Gloag, Turnbull, and Whitchurch (2015), Heylighen (2015b, 2016), Marsh and Onof (2008), and Weinbaum (2018).

On the most abstract level, computation is the process of manipulating data. Traditionally computation has been associated with the concept of the Turing machine (Turing, 1937) which describes it in terms of two abstract components: *process* and *memory*. According to Turing's model, a process *reads* data from a certain place in the memory, *manipulates* it according to provided instructions and *writes* it to a certain place in the memory. By combining different instructions and carefully ordering places (addresses) of the memory where the data is read/written, any kind of data manipulation, no matter how simple or complicated, can be achieved. Stigmergic computation can also be described along these lines – it is composed of more than one (typically a great number of) individual instructions/processes ("ants") which read and write to a shared memory ("environment"). The most important difference between these two models is that in the case of stigmergy, instructions are not strictly ordered and places / addresses in memory are not strictly defined – they "self-organize" during the computational process itself.

It seems perfectly appropriate to attempt to explain the emergence of collective control within the hierarchy of elementary feedback units of perceptual control theory (see page 43) in terms of stigmergic computation and, first and foremost, the emergence of hierarchy (see Section 2.2.3) – which, after all, is just a pattern of connectivity. The concept of *stigmergic computing* will be developed further in Section 4.5.

Note, however, that shared memory as an external logical or physical entity is not needed for stigmergic computation to occur. The function of shared memory is the ability to pass information indirectly (and perhaps probabilistically) to "unknown" receivers – whoever is passing by and is willing to pick up the information left in the medium. This kind of communication is usually contrasted with direct communication, where each process sends the information to one or a few known recipients. Yet the same effect can be achieved by broadcasting information to any recipient which is capable of receiving it, for example via the mechanism of *spreading activation* (see Section 2.3.1).

2.2.5 Guided self-organization

Guided self-organization is the multidisciplinary area of inquiry aimed at *finding ways to guide the processes that seemingly spontaneously self-organise towards desirable outcomes*. In simple terms, it is a field of ongoing research into engineering the evolutionary development process resulting in defined complex adaptive systems (in a functional or structural manner). It is considered among the most complex engineering tasks (Prokopenko, 2009, 2014). The research within the domain of guided self-organization draws on methods from computational, physical, biological domains, information theory, theory of computation, dynamical systems, machine learning, evolutionary biology, artificial life, statistical mechanics, thermodynamics, and graph theory – and therefore touches many aspects of the open-ended decentralized computing concept developed by this work. We build on these parallels along the way.

Based on what we covered so far, we can formulate the following principles, which will guide the design of the computational model in Chapters 4 and 5:

- The emergence of collective control is equivalent to the stabilization of communication patterns among independent elementary units;
- Together these patterns define a structure which gives rise to a function of the overall system and constrains emergence of further structures and functions;
- By that, the progressive determination of structure and function is achieved in an evolutionary developmental way.
- Such systems cannot be directly controlled, but guided via exerting influence on a progressive determination process, which is a complex engineering task and the research domain.

2.3 Insights from cognitive and neuro science

2.3.1 Spreading activation

Spreading activation is the mechanism of information propagation in a network, e.g. cascades of neural action potentials in a biological brain. The term was introduced in the 1960s by Ross Quillian in the context of human semantic processing and computer simulations of memory search and comprehension of language (M. Collins and Loftus, 1975). Since then it has become the main algorithm for working with network data structures: associative, biological, artificial neural networks as well as semantic networks and graphs (Baronchelli et al., 2013; Gouws, Rooyen, and Engelbrecht, 2010; Heylighen, 2008; Heylighen, 2001a, 2002a; Heylighen and Bollen, 1996; Rodríguez, Gayo, and Pablos, 2013).

In simple terms, the mechanism works by activating one or more nodes in a network and then propagating this activation to neighbouring nodes via associative or semantic links. The quantity of activation (or action potential in biological neural networks) is modulated by the strength of links between nodes or semantic parameters in case of semantic networks (Heylighen, 2009b; Rodriguez, 2011). When a node gets activation from several links, the activations are summed and the sum is propagated further. In this way a mechanism can learn the structure of a network, or a

particular aspect of it. Dynamic networks can also be modelled using this algorithm by allowing activations to change link strengths (in biological neural networks) or semantic properties of nodes and links (in semantic networks and graphs). Importantly, the algorithm naturally allows for parallel and distributed processing.

2.3.2 Ecological rationality

Ecological rationality is a concept introduced by Gigerenzer (2008) which frames rationality as a match between mind and environment rather than an ideal human reasoning and probabilistic inference. The concept reflects and builds on the view that "intelligent behaviour in the world comes about by exploiting reliable structure in the world – and hence, some of intelligence is in the world itself" (Todd and Gigerenzer, 2012). Simon (1990) framed a similar principle in terms of the image of scissors, saying that the rational behaviour of all physical symbol systems is shaped by a pair of scissors whose two blades are the structure of task environments and the computational capabilities of the system – and both have to be taken into account when envisioning an embodied intelligent being. The perspective of evolutionary epistemology (Section 2.2.1) is even more radical, basically claiming that all knowledge is nothing else than internalization of the "intelligence of the world".

Cognitive psychology therefore can be approached as the study of computational capabilities of a certain embodiment of intelligence facing diverse tasks. These capabilities may involve logical reasoning, statistical sampling, probabilistic inference and heuristics whereas none of these modes fit the ecological purpose automatically – i.e. they should be dynamically selected depending on the context. Ecological rationality emphasizes the intractable structure of the world and importance of heuristics for interacting with it. Heuristics deal with uncertainty and limits of computational capacity by "smartly" ignoring part of the information about the environment (see the principle of *selection for relevance* on page 29). It does not try to achieve optimal results, as do optimization procedures or algorithms, but rather satisfy – i.e. achieve "good enough" results.

By emphasizing relations with an environment, ecological rationality blurs the borderline between perception and cognition. Importantly, it allows cognition to be viewed in terms of non-deterministic and context-dependent computational processes (see Section 4.2.3). Furthermore, it blurs the boundary between "system" and its "environment" by positing that the intelligence and cognitive capabilities of an agent cannot be analysed or understood without reference to its environment. A philosophical system that addresses these issues in a rigorous way and thus provides a strong conceptual background is the theory of individuation discussed in Section 3.2.

2.3.3 Rate distortion theory

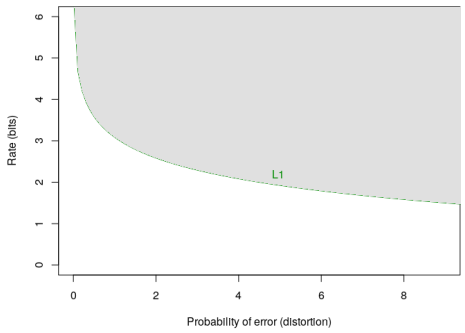
A framework that proposes practically applicable formalization and definition of important aspects of ecological rationality in mathematical terms is the *rate distortion theory* (Berger, 1971) – a major branch of information theory that provides theoretical foundations for lossy data compression. The relevance and value of the

rate-distortion theory is that, as Sims (2016, p. 13) formulates, it "combines the central elements of both information theory and decision theory, and is uniquely situated for explaining biological computation as a principled, but capacity-limited system". It provides an excellent framework for discussing the fundamental trade-off between comprehensibility and information loss (discussed on page 31). Moreover, rate distortion theory supports the conceptual approach to the general process of *understanding* as a lossy information compression (see page 32) – an approach that I consider one of the major principles for conceiving a synthetic cognitive system.

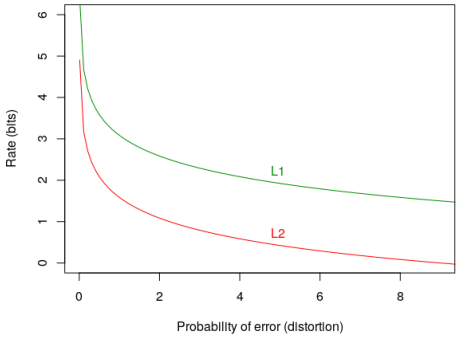
Perception is a process of extracting meaning from the noisy and uncertain environmental signals (i.e. sensory space) and choosing which ones to transmit to a decision-making process and which ones to discard. Rate-distortion theory looks to perception as a communication channel of limited capacity which necessarily implies loss of information in cases where the input rate is larger than channel capacity (which is **always** the case with the natural environment). Therefore, the goal of perception cannot be the perfect transmission, storage, or reproduction of afferent signals, but rather the minimization of some cost function subject to constraints on available capacity. The rate-distortion theory describes and formalizes this trade-off by quantifying how the minimum necessary channel capacity depends on performance requirements in terms of information transmission (ibid.).

The rate-distortion curve (Figure 2.12a) defines the ideal boundary performance of a perception–memory system (as an information transmission channel or information compression system) in terms of its *structural efficiency* given the goal (i.e. cost function) of the system. The ideal boundary performance is the theoretically maximal information transmission rate given the probability of error in the transmission channel (i.e. distortion). Considering the ecological nature of intelligence, it may be more important for the decisions of the perceptual–cognitive system to be "good enough" in a specific situation than "efficient" or "optimal". The crucial distinction is that the "good enough" decision is subject not only to the internal neural architecture of the embodiment in question, and the costs and constraints it imposes, but also to the goals of the agent it represents and the environmental situation at the moment of the decision. Rate-distortion theory represents the perceptual goal of a system in terms of a cost function which formalizes the outcome of the *selection for relevance* aspect of modelling – depending on the chosen cost function, the ideal boundary performance defined by rate-distortion theory may differ (see Figure 2.12b).

When applied to human cognition, rate–distortion theory describes three critical components that limit human perceptual memory: (1) information-theoretic limits on memory capacity, (2) mismatches between ecological statistics and implicit statistical learning, and (3) mismatches between task-defined and implicit cost functions. Importantly, it explicitly considers the effect of a system's goal on the measure of its performance – allowing for thinking in terms of multiple goals without presupposing a single overarching "objective" one. The aspect that the theory does not encompass, however, is the mechanism of choosing a concrete goal in a concrete situation – i.e. selection for relevance aspect of modelling.



(A) Indicates the optimal trade-off between probability of error (distortion) and rate of information transmission for a given cost function L1; any real channel can operate only above the optimal curve in the shaded area of the plot. The distance of an actual system from the optimal curve measures the *structural efficiency* of the channel.



(B) Indicates two different cost functions L1 and L2; in a complete open-ended model, the cost function represents the goal of the perceptual-memory system and is a parameter which can vary depending on the environmental situation.

FIGURE 2.12: Rate-distortion curve. Adapted from Sims (2016).

2.3.4 Coherence

As defined by Thagard (2002, p. 34), coherence can be understood in terms of maximal satisfaction of multiple constraints in a system or theory. A system is comprised from a set of *elements* or representations, such as concepts, propositions, perceptions, images, goals, actions, etc. Elements are related to each other with the *relations* of coherence; if two elements cohere (i.e. explain, deduce, facilitate, associate, etc. with each other), then there is a positive constraint between them; if elements incohere (i.e. are inconsistent, incompatible or negatively associated), then there is a negative constraint between them. The degree of coherence is obtained by dividing a set of elements and constraints into accepted and rejected subsets. A positive constraint can be satisfied either by accepting both elements or rejecting both elements. A negative constraint can be satisfied only by accepting one element and rejecting the other.

Two notes are of relevance here. First, the coherence problem has clear relations to the maximum satisfiability problem (MAX-SAT) in computer science and could be considered a computational problem, albeit requiring a proper representation. Second, the coherence problem naturally lends itself to being represented in a graph form (i.e. in terms of elements and their relations) – an important property which we will expand on later (see Section 5.2 Graph computing on page 130).

While the coherence problem as defined by Thagard allows for a fuzzy or continuous degree of coherence (e.g. calculated as the proportion of satisfied constraints with respect to all constraints in the system), it allows only for discrete constraints. Yet much more realistic would be a system with fuzzy constraints ranging from fully positive to fully negative. It may be that the difference of using discrete or fuzzy constraints in solving coherence problems is precisely the difference between symbolic and sub-symbolic processing and computing. The insurmountable gap between symbolic and sub-symbolic cognition stems from the fact that all perception is sub-symbolic, but in order to apply reasoning to it, the fuzzy sub-symbolic relations have to be converted into discrete and unambiguous ones. During this process, some information about the system is necessarily lost.

2.3.5 Integrated information

Integrated information is the central concept of the mathematical theory of consciousness introduced by the neuroscientist Tononi (2004). The thesis of integrated information theory (IIT) is that the degree of consciousness of a system is correlated to the measure of integrated information Φ. Intuitively, Φ measures how much information is generated in the system by the virtue of being an irreducible whole rather than a collection of independent components. The basic idea is to estimate how much information is generated by the components when considered independently and compare it to the amount of information generated by the system as a whole (Tononi, 2008). This can be understood as a measure of "non-decomposability" or "irreducibility" of a system to its parts. Note from this definition that information integration is actually a *measure of complexity* of a complex adaptive system composed of interacting heterogeneous components.

Conceptually, IIT aligns well with the concept of open-ended intelligence and individuation of intelligence (Weinbaum, 2018; Weinbaum and Veitas, 2017a; Weinbaum and Veitas, 2017b) in that the information integration measure can be used to account for the "degree of individuation" of the system (the concept will be explained later on page 64 in terms of distinction between preindividual, *fluid individual* and *fixed individual*). Concretely, IIT uses the notion of *intrinsic information*, which is defined as "a difference that makes a difference" from the perspective of the system itself, i.e. without relying on an external observer. In that it differs from Shannon's notion of *extrinsic* information (Oizumi, Albantakis, and Tononi, 2014) and, furthermore, allows for information to individuate during the development of a system, which is a fundamental aspect of the developmental systems theory (see page 37).

2.3.6 Human cognitive development

The concept of cognitive development has been defined in the field of psychology as "the emergence of the ability to understand the world" (Schacter, Gilbert, and Wegner, 2010, p. 447). Traditionally it is mostly associated with the child development stages proposed by Jean Piaget but can be also applied to describe sense-making by an individual throughout its whole lifetime as proposed by Kegan (1982). Piaget originally contended that children pass through four eras of development - sensimotor, prelogical, concrete operational, and formal operational - which can be further subdivided into stages and substages (Kohlberg and Gilligan, 1971; Piaget, 2004). Kegan also propounded that Piaget's and some later cognitive development theories generally describe recursive subject and object relationships when the subject of a previous stage becomes an object in the next stage, to which he refers as an "evolution of meaning". Subject in this context means whatever is perceived as part of self while object is part of the environment. Therefore cognitive development can be understood as *an ongoing balancing of subject – object relations and interactions across the emerging boundary of an individual towards increasing cognitive complexity* (Weinbaum and Veitas, 2017a). This recursive process progressively defines a boundary of an individual – a psychic differentiation of *self* from the *other* (Kegan, 1982, p. 24) which constitutes the differentiation between agent and environment.

For further clarification of our understanding of cognitive development as individuation and the benefits of such an approach, let us examine a schema of Era I of early cognitive development as formulated by Piaget (Table 2.1). It is clear that every

Stage 1.	Reflex action.
Stage 2.	Coordination of reflexes and sensorimotor repetition (primary circular reaction).
Stage 3.	Activities to make interesting events in the environment reappear (secondary circular reaction).
Stage 4.	Means/ ends behavior and search for absent objects.
Stage 5.	Experimental search for new means (tertiary circular reaction).
Stage 6.	Use of imagery in insightful invention of new means and in recall of absent objects and events.

TABLE 2.1: Era I (age 0-2): The era of sensorimotor intelligence. Adapted from Kohlberg and Gilligan (1971, p. 1063)

subsequent stage builds upon the previous one and together they seem to form a hierarchy. It seems, however, that cognitive development theorists and practitioners, including Piaget, agree that stages in cognitive development overlap, occur in parallel or get manifested later in the maturation process. Therefore we can approach the process of cognitive development as both a sequence of stages and a continuum. In Chapter 3 we will see that a developmental continuum punctuated by distinct stages is also supported by understanding cognitive development as a case of individuation. The appearance of stages of cognitive development seems to be better understood in terms of dynamic products of individuation or "evolutionary truces", as Kegan calls them, rather than pre-defined "milestones".

2.3.7 Enaction

The enactive approach treats cognition as the adaptive process of interaction between an agent and its environment. Actually, the distinction between agent and environment is constituted by the interactions themselves. A *cognitive system* can be seen as a complex adaptive system which is an organized network of interactive sub-processes (De Jaegher and Di Paolo, 2007, p. 3) that together realize a network of objects and their relations as they are perceived in the world.

A cognitive system cannot form itself separately from the matrix of interactions with other entities within a larger population. In terms of social psychology this principle is informed by a perspective that minds exist only as social products (Summers, 1994, p. 328). Relationships and bonds with other entities of the population are part of the cognitive system and thus define its identity on equal terms with internal relationships and structures. Therefore, the mental states of an individual are not established prior to the interaction, but are shaped, or even created, during its dynamics. Di Paolo and De Jaegher (2012) describe these dynamics as participatory sense-making and propose the *interactive brain hypothesis* which "describes an extreme possibility, namely that all social brain mechanisms depend on interactive elements either developmentally or in the present, even in situations where there is no interaction" (ibid., p. 5).

Also, in some forms of psychotherapeutic theory and practice (e.g. Gestalt and the interpersonal approach to psychoanalysis), certain interactions or situations which are normally considered external to an individual are actually an integral part of its sense-making processes. An individual enacts itself in its social milieu rather than merely using internal representations, plans or theories of mind or even perceptual routines existing prior to the interaction.

Edelman and Mountcastle (1982) define "world inputs" and "self-inputs" to differentiate between interactions across and within the boundary of a neuronal group. Weinbaum and Veitas (2017a) extend this principle from the context of neuronal groups to networks of cognitive agents. An individual is defined as a totality of both types of interactions while the proportions of them may differ at different periods (see Section 3.3.3). Likewise, Brender (2013) relates cognition to bodily movement in an environment: "we cannot conceive of a difference in nature except by reference (implicit or explicit) to a bodily movement that would reveal this difference. [...] Thus we cannot give an account of nature that is not an embodied account, that does not take up the point of view of a moving body situated within the nature it describes".

2.3.8 Sense-making

Sense-making is one of the components of the enactive approach to mind and cognition (De Jaegher and Di Paolo, 2007). Weinbaum and Veitas (2017a) frame cognition as a process of individuation within the scope of what is referred to by Piaget (2004) as "genetic epistemology". A psychologically-oriented definition of sense-making is the following: *sensemaking is a motivated, continuous effort to understand connections (which can be among people, places, and events) in order to anticipate their trajectories and act effectively in relation to them* (Klein, Moon, and Hoffman, 2006, p. 3). From the perspective of dynamics of the cognitive system, sense-making is a continuous effort to form a network of connections and objects as they are perceived in the world. The enactive approach implies that cognition and sense-making are seen not as something that happens inside clearly defined boundaries of the cognitive system but as the product of interactions (McGann, 2008) across emerging boundaries: “sense-making establishes a perspective on the world with its own normativity, which is a counterpart of the agent being a center of activity in the world” (De Jaegher and Di Paolo, 2007, p. 4). Or, as Brender (2013) puts it, "the organism and its world grow together dialectically, each driving the other to become more articulated and determinate through its own increasing determinacy. This is the growth of sense: the self-articulating field of differences that make a difference to the organism" (ibid., p. 271).

Sense-making has the following notable aspects:

- **Identity and identification**. A prior notion of an entity ‘which makes sense’ seems to be needed, but in the framework of open-ended intelligence it is not the case: the identity of cognitive agents is created during the process.
- **Enaction**. According to Clark (2012) perception is an action where an agent produces a stream of expectations and then corrects its own model according to incoming information. Therefore the primary component of sense-making is an action: an agent acts upon the environment, catches the "reflection" or response and updates the internal representation of it.
- **Reflexive**. Sense-making is a two-way interaction between the individual and its environment across the boundary being created during the same process: any agent’s examination, modeling and action "bends" the environment and affects the perception of and further decisions by that same agent. The property of reflexivity of the system captures these mutual influences of networks of processes across the boundary of an agent.

- **Participatory aspect**. As noted by Di Paolo and De Jaegher (2012), "mental states that 'do' the understanding and the ones to be understood are not fully independent or established, but are instead affected, negotiated, and even created as a result of interaction dynamics" (ibid., p. 4). They describe the set of possibilities arising from these dynamics with the notion of *participatory sense-making*, emphasizing its social aspect. In Section 3.3.2 we extend the social aspect of sense-making across multiple scales with the general framework of individuation of interacting population of elements.

2.3.9 A worldview

The essence of the sense-making process is already encoded in the word itself – it is an active "making" of a "sense" or "meaning" by an observer – a cognitive agent. The concept does not overlook the fact that sense-making is based on extracting information about observable patterns in the system (the world, self and others) being perceived. But, at the same time, it emphasizes that it is the observer who decides what the *significant* patterns are to extract from the data about a system or phenomenon. Sense-making is rooted in the enactive approach to cognition (Section 2.3.7) which puts the concept in a larger context, first of all, entailing the individuation of the very agent which performs sense-making[11,12].

The process of sense-making begets a *worldview*. Importantly, the relationship of the sense-making and a worldview is a reflexive one – the worldview of an observer determines significances which then influence the sense-making process of the same observer. The concept of a worldview is a rich and multi-dimensional one (see Vidal (2008) and Vidal and Dick (2014) for an in-depth discussion and references). It can be understood as *a Gestalt perception – a unique and integrated cognitive structure – held individually or collectively in relation to self, others, society, and the cosmos at large* (Markley and Harman, 1981; Veitas and Weinbaum, 2015). With respect to the social system we live in, each worldview includes our aspirations, the views on "natural tendencies" and "trends" of the system, related possibilities for the future, and approaches to the appropriate modes of social governance. Each of these aspects is based on a combination of sense-making perspectives which may be overlapping, incompatible or even mutually exclusive. For example, individuals or collectives may prefer exploration, growth and development of persons, society and life in general, or, alternatively, stability, safety and preservation. Often such preferences cannot be accommodated within a single value system and represent different perspectives towards the same phenomena. The society of mind is therefore the multiplicity of interacting embodiments of worldviews, representing different value systems and points of view. No single value system or worldview can be considered dominant or "objectively" better, while the resilience and growth of the global system depends on the mode of interaction among many worldviews rather than any isolated properties of one of them.

[11] We employ the simplification of a well defined observer - observed distinction (i.e. agent - environment) at this point mostly for didactic purposes. Actually, the distinction between observer and observed itself individuates during the process of synthetic cognitive development (see Section 3.3.3 on page 78). For the in-depth analysis of the individuation of agent-environment boundary, please refer to Section 2 of Weinbaum and Veitas (2017b).

[12] For an in-depth definition of sense-making concept, please refer to Section 1.3 of Weinbaum and Veitas (2017a).

2.4 Summary of the chapter

This chapter is a condensed view of the modern quest to understand intelligence with a perspective informed by the philosophy of individuation and open-ended intelligence – the subject that will be comprehensively discussed in Chapter 3. The purpose of such a layout is, first of all, to show how established theories and modes of thinking within broadly multidisciplinary domains raise issues from their own perspectives which we are going to address with the metaphysical framework of open-ended intelligence on a highly integrated and abstract level. It invites the reader to appreciate the further introduced philosophical framework as a fertile conceptual lens allowing these issues to be seen in a perspective which permits new ways of theoretical understanding as well as the pragmatic research and development of intelligent systems.

First, we introduced AI research perspectives, or currents, and position them on the axis of *freedom and constraint* according to how much change they allow for an intelligent system (or any human-designed complex adaptive system). Second, we went on to introduce the evolutionary perspective to the process of becoming intelligent in terms of biological and epistemological evolution, interactive co-emergence of structural and functional hierarchies in complex systems, and information-theoretical interpretations. Third, we discussed frameworks and insights from cognitive science, neuroscience and other interdisciplinary domains. Altogether, this chapter presents main concepts, techniques and currents of theoretical and practical thought that will allow us to consolidate the model of open-ended decentralized computing in Chapters 4 and 5.

Concretely, in this chapter we have formulated a list of principles and broad requirements for the cognitive architecture:

i. Such architecture should allow for the emergence of identities of higher order cognitive agents from the interaction of lower level heterogeneous components.

ii. It has to account for at least partial ability of the emergent cognitive agents to perform the selection for relevance operation of modelling the sensory input of their environment.

iii. An evolutionary process implies the appearance of order (i.e. asymmetry) out of disorder (i.e. symmetry) in terms of emergence of functional and structural hierarchies as control structures.

iv. A cognitive system, as a complex adaptive system, balances between disorder and order, which are equivalent to internal symmetry and asymmetry.

v. Complex adaptive systems are such because they do not have a single or stable control structure, but collective control structures (e.g. heterarchies). The emergence of such collective control is equivalent to the stabilization of communication patterns among independent elementary units which make up the system.

vi. Stable or at least partially persistent patterns define a structure which gives rise to a function of the overall system and mediates emergence of further structures and functions. By that, progressive determination of structure and function is achieved in an evolutionary developmental way which underlies the self-modifying nature of a cognitive architecture.

The next chapter introduces and discusses the open-ended intelligence framework and the philosophy of individuation which binds the still somewhat isolated principles and requirements for the computational model into an integrated theory.

Chapter 3

Open-ended intelligence

The chapter is based on the following published papers:

Weinbaum (Weaver), D., Veitas, V. (2017). Open ended intelligence: the individuation of intelligent agents. *Journal of Experimental Theoretical Artificial Intelligence*, 29(2), 371–396. https://doi.org/10.1080/0952813X.2016.1185748

Weinbaum (Weaver), D., Veitas, V. (2017). Synthetic cognitive development: where intelligence comes from. *The European Physical Journal Special Topics*, 226(2), 243–268. https://doi.org/10.1140/epjst/e2016-60088-2

Veitas, V., Weinbaum, D. (2017). Living Cognitive Society: A "digital" World of Views. *Technological Forecasting and Social Change*, 114, 16–26. https://doi.org/10.1016/j.techfore.2016.05.002

3.1 Introduction

In Section 2.1 we identified the major AI research perspectives and types of intelligence they explicitly or implicitly assume, and positioned them on the axis of *freedom and constraint*. Recall that the *freedom and constraint* continuum ranks concepts of intelligence according to how much variety and change they allow. Intelligence manifested as a deterministic computer algorithm (or a rule-based reasoning engine) occupy the rightmost part of the continuum, allowing no possibility for the algorithm to "decide" its course of action – since all its goals and behaviours are encoded into rules that are pre-defined before the algorithm starts running. Open-ended intelligence, occupying the leftmost part of the spectrum, allows for any conceivable behaviours, choices of goals and forms of embodiment powering these choices. The most interesting manifestations of intelligence are arguably positioned in between these two extremes, embodying a unique synthesis of both freedom and constraint able to comprehend, operate and persist in specific environments. The synthesis of freedom as an unconstrained possibility space and constraint as a single actuated possibility is a *process* that allows and guides the evolutionary development of an individual. The goal of this chapter is therefore twofold:

- First, introduce the philosophical framework which allows and promotes looking at all manifestations of intelligence as snapshots (or projections) of the fluid

process that open-ended intelligence is. A complete account of open-ended intelligence as a metaphysical framework is developed by Weinbaum (2018).

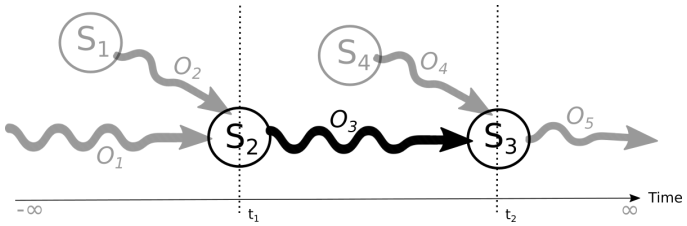
- Second, emphasize, describe and operationalize the abstract mechanism of open-ended intelligence – the process of becoming intelligent – in terms of progressive determination of constraints. This mechanism allows for concrete types of intelligence to be crystallized from the unbounded space of possibilities and manifested in observable forms. The operationalization should allow for a pragmatic perspective that guide a design inquiry into engineering artificial general intelligence. The pragmatic aspects will gradually gain more weight in Chapters 4 and 5.

At the roots of our conceptual framework, based on the open-ended intelligence philosophy, is the principle that *intelligence is a process which creates order from disorder*. Compare this principle to the more conventional way of thinking that intelligence is a *process of finding the correct order* of nature and environment – which starts with the assumption that the correct order exists in the first place. The line of thinking within the scope of this work avoids the question of whether nature actually has inherent order and structure. Rather, it adopts the epistemological position while taking into account that any real world intelligence is necessarily embodied, situated and hence constrained by limited resources. It means that from the perspective of an entity that is in the process of becoming intelligent, the order of the universe (or the structure of this entity's immediate environment) is subjectively unknowable and has to be *created* within the cognitive system of that entity. In order to understand what intelligence is and how to model it, it therefore makes sense to assume a disordered universe and entities (agents) trying to make sense of it by creating order: individually in their own cognitive systems and collectively through their shared local environments.

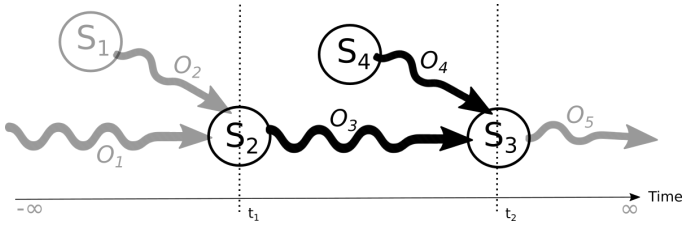
The conceptual shift required by this attitude is well described by the three different avenues of approaching the question *how does form arise?* formulated by Oyama (2000): "preformationist", "interactionist" and "constructivist interactionist" (see page 37). The first two assume *a priori* sources of information. These sources get morphed into concrete forms by an evolutionary process that is understood essentially as a process of information transfer (Zenil et al., 2012). The "constructivist interactionist" avenue does not assume the existence of *a priori* information preceding the process of evolution and development, since the information that describes forms co-develops along complex processes that give rise to these forms. We are taking the "constructivist interactionist" avenue as cardinal and the first two as special cases of it. An attempt to visualize the differences and relations among the three avenues is provided in Figure 3.1.

With a considerable concession towards simplification, Figure 3.1 illustrates how the "preformationism" and "interactionism" are seen as reductions of the "constructivist interactionist" schema[1]. The "preformationist" (3.1a) attitude amounts to viewing a process of development within a bounded time frame $[t_1, t_2]$ and therefore observing already individuated structure S_2 *as-if a priori* to the operation O_3 and operation O_3 *as-if* goal directed with respect to the informational content of the structure

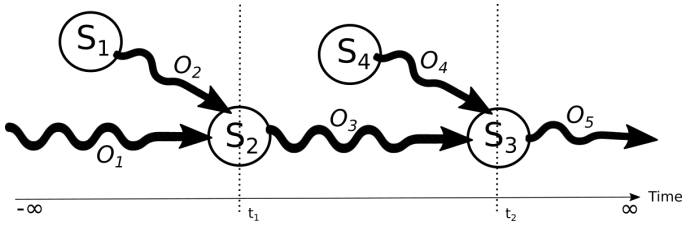
[1]The didactically necessary simplification involves omitting the aspect of the progressive determination principle that requires operation O_i to be prone to change depending on the immediate structure S_i on which it operates. This simplification allows the point we are making here to be emphasized. For a more complete treatment of progressive determination, see Section 3.2.4 on page 74.



(A) "Preformationism" is visualized as $S_2 \rightarrow O_3 \rightarrow S_3$



(B) "Interactionism" is visualized as $S_{2,4} \rightarrow O_3 \rightarrow S_3$



(C) "Constructivist interactionism" is visualized as $...O_1 \rightarrow S_{1,2,4} \rightarrow O_{3,4} \rightarrow S_3 \rightarrow O_5...$

FIGURE 3.1: Approximative illustration of relation and difference of three avenues for approaching the question of *how does form arise?* (see page 37) when viewed from the perspective of *progressive determination* (Section 3.2.4). S_i denotes informational content of an intermediate structure observed in a complex process; O_i denotes intermediate operations which progressively determine structures. Note that all three visualizations suggest different views of the same underlying complex process.

S_3. The "interactionist" attitude (3.1b) follows the same schema yet allows interactions of multiple operations $O_{3,4}$ on multiple *as-if a priori* structures $S_{2,4}$ that lead to the *as-if* goal state S_3. "Constructivist interactionism" does not look at the process of development in terms of bounded time, and therefore informational content of all intermediate structures $S_{1,2,3,4..}$ progressively develops via interaction with operations $O_{1,2,3,4,5..}$ and does not lead to any goal state. Since in all real world situations we have to consider time and space constraints, the "preformationist" and "interactionist" attitudes in the great majority of pragmatic cases "just work". Yet when dealing with complex self-organizing systems with a large degree of non-determinism these attitudes reach their limits. In this thesis, above all in Chapter 4 and Chapter 5, we will try to see how "constructivist interactionism" can help to deal with these limits while still keeping a pragmatic attitude.

Let us now see how the described conceptual shift applies and informs the quest for understanding intelligence and designing machine intelligence. It seems customary and common-sensical to posit a single (albeit possibly complex) criterion and position intelligence of humans, other known biological organisms, and artificial or natural (yet maybe not terrestrial) beings along a single dimension (see Figure 3.2). A single dimension is basically conceived to measure how well different types of intelligence fare in searching for structure and order of their environments.

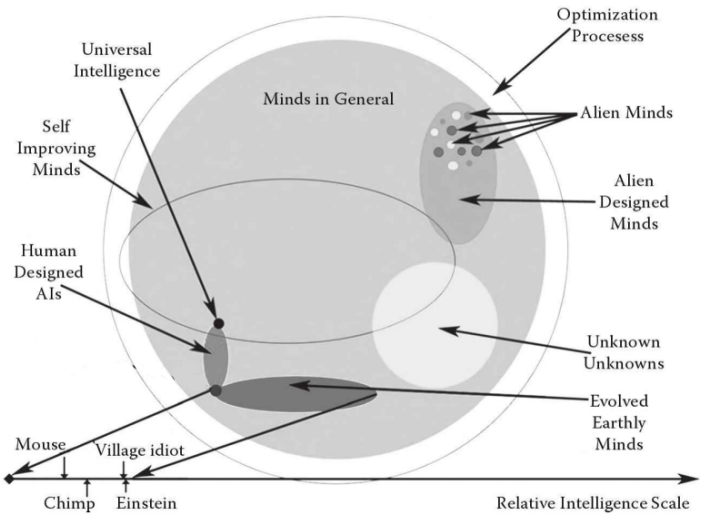


FIGURE 3.2: The universe of possible minds (Yampolskiy, 2015) positioning different kinds of mind and intelligence in an abstract "optimization space" allowing them to be projected on a single axis, first proposed by Bostrom and Ryan (2015).

This almost archetypical view of intelligence implies quite a few premises that are often taken for granted. The first premise is that reality has an implicit structure (Zenil et al., 2012). The second premise is that intelligence has evolved for and therefore is an instrument and means-to-ends for the survival of an organism. The third and most important premise is that the environments (sensory spaces) to which intelligences adapt by interacting *do not change* due to the interaction (i.e. are non-reflexive). It does not take many logical steps to conclude from these assumptions that intelligence is a complex convergent optimization process for the sake of survival that tries to adapt to the actual structure of reality and where optimization criteria are extrinsically and objectively defined. It also promotes "understanding" intelligence in terms of comparing and ranking different embodiments by how well they perform. Following this logic, the "higher" the intelligence is, the more capable it should be at finding and adapting to the structure of reality faster, better and more efficiently.

A number of theories and scientific perspectives, introduced in Chapter 2, challenge at least some of these premises in important ways. Evolutionary development (see page 35), niche construction (page 36), developmental systems theory (page 37) and perceptual control theory (page 43) all show complex relationships between organism, a cybernetic system and environment that goes beyond adaptationist paradigm. Additionally, ecological rationality (page 50) proposes a way to think of intelligence as subjective in relation to immediate environments and selective sampling of effective sensory spaces. The rate-distortion theory (page 50) points to the information-theoretical account of selection for relevance aspect in a perception-memory system where an organism has a say as to which part of sensory input it discards in order to make sense of it. The concept of open-ended intelligence provides a framework of distributed systemic cognition including an integrated philosophy of how to think about intelligence that transcends adaptionist paradigms and goal directed behaviour (see Figure 3.3). In this chapter we introduce the main tenets of this philosophy and discuss its aspects relevant to the design inquiry into understanding (and engineering artificial) intelligence.

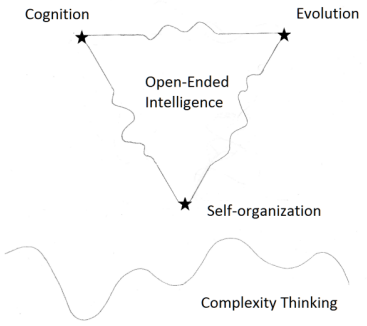


FIGURE 3.3: The scaffold of a concept of open-ended intelligence with complexity thinking as a ground and evolution, cognition and self-organization as pillars embedded into the ground. Intelligence gets bootstrapped through dynamic interaction of all these elements in an open-ended way. Adapted from Weinbaum (2018).

3.2 Theory of individuation

The philosophy of individuation by Simondon (1980, 1992, 2005, 2009) opposes the hylomorphic schema which posits the dichotomy of form and matter and sees the form, the matter, the objects and the relations among them individuating together without any primary principle defined prior to this individuation. It understands an individual from the perspective of the process of individuation rather than the other way around. An individual is a metastable phase in a process and is always in possession of not yet actualized and not yet known potentialities of being:

> Individuation must [..] be thought of as a partial and relative resolution manifested in a system that contains latent potentials and harbors a certain incompatibility within itself, an incompatibility due at once to forces in tension as well as to the impossibility of interaction between terms of extremely disparate dimensions (Simondon, 1992).

An individual that is brought forward via this process is never a complete entity but rather an intermediate state of becoming which stands out from its environment just enough to be identified by an observer and possibly by itself. Such an individual emerges only via relations to its immediate environment, which consists of other individuating entities (together constituting a *preindividual*):

> The relation is not an accidental feature that emerges after the fact to give the substance a new determination. On the contrary: no substance can exist or acquire determinate properties without relations to other substances and to a specific milieu. To exist is to be connected. This philosophical proposition allows Simondon to establish the scope of his project: to reconcile being (*l'être*) and becoming (*le devenir*) (Chabot, Krefetz, and Kirkpatrick, 2013, p. 77).

Importantly, Simondon's theory of individuation, while being an abstract ontological framework, at the same time promotes what is called ''concretization" - the explanation of the emergence of observable and graspable objects and relations in the physical, biological, cognitive and socio-technical evolution and development (Veitas and Weinbaum, 2017; Weinbaum and Veitas, 2017a). Simply speaking, "concretization" allows us to approach the very process of emergence of order from disorder in an abstract way. A schematic image of this process is depicted in Figure 3.4. It visualizes a population of independent interacting agents, initially uncoordinated

but still exchanging signals among themselves – a *preindividual*. Over time, some interactions may grow more frequent and possibly reciprocal, constituting to stronger, but not yet fully persistent links between some elements. Thus a *fluid individual* can be observed in the population. If certain links become persistent, a *fully formed individual* with a definite structure and a boundary, separating it from other members of population, becomes observable.

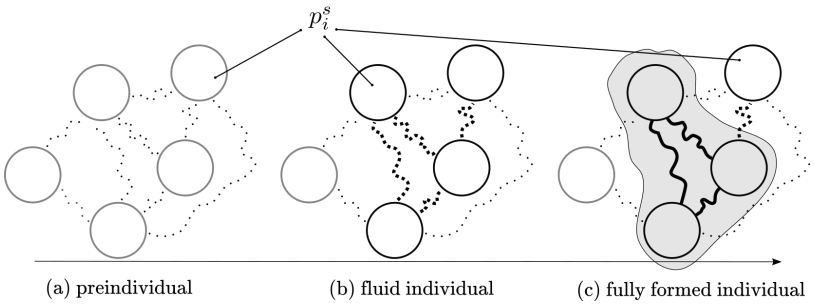


FIGURE 3.4: Image of concretization process illustrating how *(a) preindividual* consisting of a population of heterogeneous non-coordinated agents morph into *(b) fluid individual* and then *(c) fully formed individual*. All three are but intermediate stages of the individuation process and constitute different assemblage configurations of interacting elements p_i^s in a population P_s (see also Section 3.2.4 and Figure 3.11).

The cornerstone of the theory of individuation is the philosophy of information, which extends the classical information theory of Shannon (1948). Other direct descendants and components of the philosophy of individuation are theory of assemblages, metastability and the notion of progressive determination, all central to and required for understanding open-ended intelligence – the process of individuation of intelligent agencies. These aspects are introduced in the following sections.

3.2.1 Philosophy of information

Recall that Shannon's information is defined as the reduction of uncertainty of a receiver due to getting a sender's message. This notion carries an immense pragmatic value in thinking and designing cybernetic systems where information is *extrinsically signified*. Signification of information is the operation of attaching a meaning to it that is shared and makes sense for both sender and receiver. Without signification, there cannot be a directional information transfer, since a sender can never "know" how and if at all a receiver will be affected by the transmitted signal.

The mathematical theory of communication considers (ibid.) an elementary unit of information, the meaning of which is assumed to be universally defined outside the theory and therefore there is no need to consider the problematics of its signification. This allows information to be defined in terms of intrinsic entropy of a message – without explicit reference to sender and receiver. Thus defined Shannon's information is the expected information contribution of a message to the reduction of uncertainty, which is an entropy measure $H = -\Sigma p_i log_n(p_i)$, where p_i is the probability of occurrence of the i^{th} possible value of the source symbol and n is the measure of information (e.g. $n = 2$ in a case where information is measured in bits).

It is surprisingly easy to overlook the aspect of signification when considering interaction and information exchange among already individuated entities in an already individuated and defined environment – a set-up that is perfectly adequate for engineering goal-directed cybernetic systems in a stable environment. Yet the assumption of a universally defined meaning of information loses its value and power in a developmental system, where the elements of a system are not *a priori* defined. Moreover, the mathematical theory of communication actually supports the hylomorphic schema and makes it inescapable without challenging this assumption. The inapplicability of Shannon's information, which was defined explicitly to require no reference to meaning, in this context needs no further comment apart from Shannon's own words:

> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem (ibid.).

Interestingly, not long after the publication of Shannon's seminal work, MacKay (1969) noted the unfortunate consequence of this statement – that the warning has been often forgotten and interpreted as irrelevance of notion of meaning to the whole theory of information. It would be fair to say that this "unfortunate consequence" has shaped the field until now.

Therefore, Simondon first and foremost extends the information theory by requiring the signification of information to be individuated itself during the interaction between sender and receiver. Traditional notions of form, matter and information are revised by stripping them of ontological primacy with respect to each other. These notions are seen as operators of a system of tensions which fuels the process of individuation (Combes, 2013). Precisely this revision allows us to leave the hylomorphic schema and embrace the open-endedness and fluidity of the process of becoming. Oyama (2000) reconstructs almost the same revision in the domain of developmental biology and evolution in terms of the developmental system theory (see page 37).

Figures 3.1 (page 61) and 3.4 (page 64) partially illustrate Simondon's extension and its importance for the theory of individuation. In this sense, "preformationist" (3.1a) and "interactionist" (3.1b) avenues of approaching the question of *how does form arise?* correspond to the classical information theoretical perspective positing operations $O_{3,4}$ in terms of communication of the information content of structures $S_{2,4}$ without considering how this information content came to be in the first place. "Constructivist interactionism" (3.1c), on the other hand, considers that the information content of all observed structures in the process have been individuating through their developmental histories. Figure 3.4 illustrates how information individuates in the process of "concretization" in terms of a structure and a boundary of an assemblage (and individual).

3.2.2 Assemblage theory

Assemblage theory was introduced by Deleuze and Guattari (1987) and further modified and developed by DeLanda (2006) as a philosophy of society. Latour (2007)

further develops an actor-network theory for approaching and exploring assemblages of social and technical objects. Assemblage theory provides an avenue for conceptualization of a generative model of individuation. In this, assemblages are sub-networks of heterogeneous individuals that have established partial compatibility among themselves. They possess an intrinsic though metastable individuality; an individuality that does not depend on an external observer but only on the relations that have been stabilized among their internal elements. DeLanda (2006) has developed the theory as a philosophical framework explaining the emergence of scalable social entities such as personal networks, social organizations, markets, cities and nation states. General premises and concepts offered by the theory are broadly applicable to the study of societies of cognitive agents and living systems and, notably, cognitive systems themselves as coalitions of neurons and cognitive processes (Weinbaum, 2013; Weinbaum and Veitas, 2017b).

The process of "concretization" as schematically illustrated in Figure 3.4 corresponds to the emergence of assemblages from a population of independent and interacting agents (Chabot, Krefetz, and Kirkpatrick, 2013). At its original level of abstraction, the assemblage theory provides a direction towards formulating concrete mechanisms of the process of individuation and becoming, i.e. emergence of objects, systems and subsystems and their relations from an initial state of disorder. Note that while we emphasize the emergence of entities, the philosophy of individuation and theory of assemblages describe processes that do not have an *a priori* direction – they progressively develop along with the process. Both emergence and dissolution of assemblages, entities and different forms of individuals can be accounted for by the same conceptual framework. Theory of assemblages can be used for explaining non-directional processes of evolutionary development and niche construction (see pages 35–36):

> One and the same assemblage can have components working to stabilize its identity as well as components forcing it to change or even transforming it into a different assemblage. In fact one and the same component may participate in both processes by exercising different sets of capacities. (DeLanda, 2006, p. 12)

Before explaining the concrete mechanisms of assemblage theory, it is worth seeing how it integrates the Simondonian philosophy of information. Recall from Section 3.2.1 that Simondon extends Shannon's information theory by requiring emitter and receiver of a message to develop a common *signification* (i.e. agree on the meaning) of their respective signals before transmission can be considered an exchange of information. Information thus individuates due to the process of interaction and drives that process further. MacKay (1969) gives an excellent explanation of the impact of this requirement on the theory of information, why it is necessary beyond mathematical communication theory, and provides an informal description of it. The crux of his account is that the information-content of a message is defined by its meaning, which is "a relationship between message and recipient, rather than a unique property of the message alone". Shannon's information theory does not neglect this fact, but avoids the question of meaning altogether by explicitly assuming its *a priori* existence, a luxury not possible within the framework of the philosophy of individuation. Further, MacKay (ibid.) provides a working definition of meaning:

> [..] the meaning of a message can be defined very simply as its selective function on the range of the recipient's states of conditional readiness for goal-directed activity; so that the meaning of a message to you is its selective function on the range of your

> states of conditional readiness. [...] suppose, for example, someone tells you "it's raining". What happens? You may be immersed in a book, and may not feel inclined even to grunt an acknowledgement. But this does not mean that your understanding of the message has had no effect on you. If a sudden call comes for you to go out of doors, for example, you may now be ready to reach for umbrella or mac. If someone comes in, you are likely to ask whether he got wet; and so on. What has been affected by your understanding of the message is not necessarily what you do – as some behaviourists have suggested – but rather what you would be *ready* to do if given (relevant) circumstances arose. It is quite possible that relevant circumstances may never arise, so that a naively behaviouristic approach would reveal no sign that you had understood the message. It is not your behaviour, but rather your state of *conditional readiness* for behaviour, which betokens the meaning (to you) of the message you heard (ibid., p. 22,24).

Following this working definition, two types of information-content are proposed: *metrical*, which increases the reliance of the receiver on the result (used in Shannon's theory of communication) and *structural*, which enables new features to be added to the description of the state of conditional readiness and its context. Both kinds of information-content reduce the receiver's uncertainty but in very different ways. A communication beyond strictly technical cases usually includes both aspects.

Now, observe that an individual's state of conditional readiness is dependent on the structure of that individual (be it a simple system, cognitive system or a brain). In assemblage theory an individual is a collection of lower scale elements having stable information exchanges among them. So the information-content of interaction among elements–assemblages in a population is determined by the internal structures of these assemblages, while these structures themselves individuate as a result of the information-content of interaction – which is precisely the subject matter of Simondon's philosophy of information and inseparable from the mechanism of progressive determination.

Furthermore, assemblage theory builds on the distinction between internal and external relations which explains relations between scales in a scalable system – a multiplicity of recursively nested populations of heterogeneous assemblages which themselves consist of populations of yet lower level elements (see Section 3.3.2). It also develops concepts of territorialization and deterritorialization which are responsible for emergence and dissolution of boundaries that mediate the relations of individuals with the rest of the population environment.

Territorialization and deterritorialization

The notion of interaction between processes of *deterritorialization* and *territorialization* originated from the work of Deleuze and Guattari (1983, 1987) – first in the context of socio-economics of production, and then in relation to dynamical systems theory and self-organizing material systems. DeLanda (2006) applies the concept when developing assemblage theory as one of the dimensions / axes along which the specific assemblage is defined. Such a dimension delineates variable processes in which components of a system become involved. The involvement can stabilize the identity of an assemblage by increasing the degree of internal homogeneity and sharpness of assemblage boundaries – in which case it is referred to as *territorialization*. Or, it could destabilize the assemblage by decreasing its homogeneity and

blurring the boundaries – a case of *deterritorialization*. The main mechanism of territorialization is the formation of habitual repetition, providing the assemblage with a stable identity. The mechanism of deterritorialization is the breaking of habits, which effectively influences and changes an identity (Smith and Protevi, 2013). The process of "concretization" (Figure 3.4 on page 64) is equivalent to territorialization, which is further elaborated by Figure 3.5:

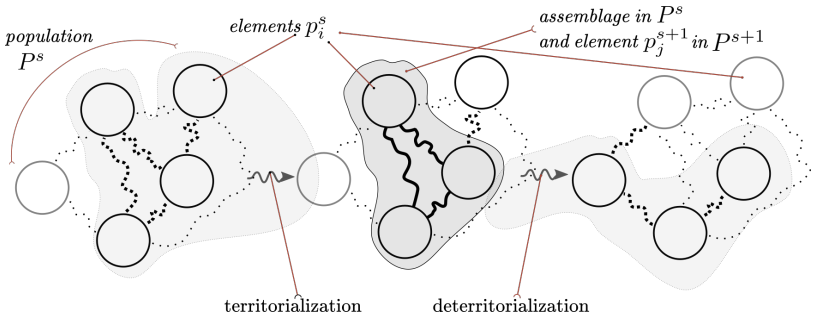


FIGURE 3.5: Illustration of territorializaton and deterritorialization dynamics within a population P^s of interacting elements p_i^s on a single dimension. Assemblages individuated in population P^s may become elements on a higher scale – a population P^{s+1} (see Figure 3.11 in Section 3.3.2 for the elaboration of the scalable model).

Note that a population P^s of elements p_i^s of Figure 3.5 can interact on many dimensions simultaneously. These dimensions do not need to be synchronized – therefore the same population of elements can integrate or disintegrate at the same time in different dimensions or form assemblages with different boundaries in each. An observer making sense of such population of interacting elements selects the relevant dimension of analysis, which is the *selection for relevance* aspect of model building (see page 29).

Boundary formation

Processes of territorialization and deterritorialization give rise to asymmetry of interactions among elements. Particular combinations of these asymmetries in turn give rise to situations where some sets of elements find themselves interacting more intensely among themselves than with other elements of a population that are not within the set. This situation is precisely what defines an assemblage – a collection of elements separated from the rest of the population by a (more or less fuzzy) boundary of lesser interactivity[2]. Boundaries effectively resolve how an assemblage of elements interacts with its environment and delineates the emergent *identity* of an assemblage as a whole. Such an assemblage with an established identity and properties – an agent – can become an element at a higher scale of individuation. Note that boundaries defining the agent–environment distinction and the relations between them are never entirely fixed. The functioning of any emergent agent is adaptive and subject to change due to alternating temporal dominance of deterritorialization or territorialization processes.

The mechanisms that are responsible for the formation of boundaries and the bringing forth of coordinated activities in a population of agents P arise primarily

[2]Lesser interactivity could manifest itself either via reduced intensity of interactions or fewer dimensions across which interaction takes place.

from the agents' intrinsic capabilities to affect and be affected by each other. Specific characteristics of these interactions, e.g. their frequency, synchronization and coherence, have a critical influence on the way agents are connected. Such influence finds its expression in the reinforcement or suppression of connections among agents and consequently on how strongly they may actually affect each other. This is how the activity of agents within P progressively determines the topological organization of the network of agents in P. The structural organization, in turn, affects the overall function of the individual agents by selecting interactions.

Recall that the Simondonian philosophy of information (Section 3.2.1) requires interacting elements or assemblages to *signify* the information – i.e. to reach mutual compatibility of signals that are considered information in their exchange. Furthermore, information is defined as signals through which elements *can affect* each other. Obviously, assemblages can be affected only by signals that permeate their boundaries, and therefore properties of boundaries determine signification of information. Since boundaries of assemblages get formed in the process of territorialization and individuation of agents, information also individuates within the same process.

Flynn (2011) provides an excellent metaphor covering subjectivity of information when discussing Merleau-Ponty's philosophy:

> Merleau-Ponty argues that the Gestalt exists for a perceiving subject; it is not a part of the world as it is in itself. The stimulus does not unilaterally affect the organism in virtue of its absolute physical and chemical properties; it becomes a stimulus only insofar as the organism constitutes for itself a vital milieu which it projects around itself. The mouse in The Metropolitan Museum of Art is affected by the crumbs of cookies on the floor, but not by the Velázquez painting on the wall. In the milieu that the mouse constitutes, the crumb is desirable and the painting does not exist (ibid.).

Finally, recall Simon's treatment of complex adaptive systems as having a property of near-decomposability where a hierarchical structure can be devised by distinguishing between the interactions *among* subsystems, on the one hand, and the interactions *within* subsystems, on the other (see page 21). Assemblage theory accounts for the emergence of architecture of complexity observed by Simon (1962) in natural physical and biological systems.

Internal and external relations

As discussed, the emergence of boundaries of an assemblage brings about the distinction between (1) relations and interactions among elements "inside" the assemblage and (2) interactions between elements "inside" and the ones "outside", which cross the boundary of the assemblage. Boundaries also allow us to identify and describe identities of assemblages in terms of how they are structured internally or how they structure their interaction with the environment. In a multiplicity of recursively nested populations of heterogeneous assemblages which themselves are collectives of lower level elements, *internal relations* are defined as relations among lower-scale elements *within* the boundaries of assemblages. *External relations* are then defined as relations among elements *across* the boundaries of assemblages - i.e. with elements of other assemblages in a population[3]. Comparative intensities and dominances of internal and external relations within a population of interacting

[3]The notions of external and internal relations, as used in this work, bear important differences from relations of interiority and exteriority, which are the fundamental concepts within the philosophy of

elements define boundaries of all assemblages in this population. A boundary of an assemblage emerges from asymmetries of interactions first as an informational membrane, selecting relevant information for an assembled individual and, only then, as a structural and topological membrane (see Figure 3.6).

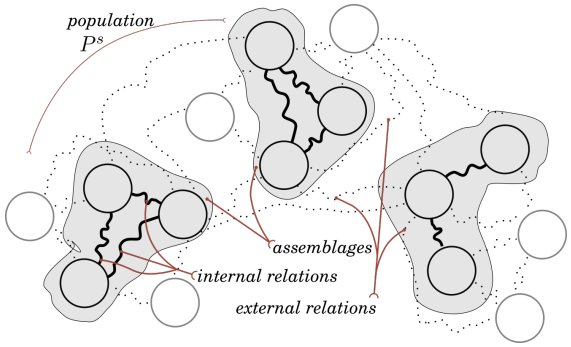


FIGURE 3.6: Internal and external relations within a population of agents and their assemblages.

Consider for example a cell and a human body as assemblages at different scales. The cell has a membrane and the human body has a skin, which, on the one hand, are topological boundaries of these assemblages. On the other hand, they are informational boundaries, defining relations to the environment that are relevant for the operation and persistence of each assemblage. The cell has ion gates on its membrane, which allows only certain ions to pass and change the internal chemistry and processes. The body has sensory organs which allow only certain types of environmental signal to be registered by an organism – which is precisely the meaning of the word "sense". Note that this adds an additional aspect to *sense-making* (see Section 2.3.8), since the development of "sense organs" is also a part of the process. Internal and external relations become more intricate in case of fluid individuals with unclear topological boundaries and fuzzy informational membranes, such as brain areas or even cognitive processes in a neural network. Here local asymmetries of information exchange between elements may be the main, if not the only, avenue for detecting and describing such fluid individuals – e.g. brain areas, social structures, colonies of insects, flocks of birds, rhizomes etc.

The problem of detecting boundaries of fluid assemblages in a population of interacting elements has been approached from the information theoretical perspective by the Integrated Information Theory, which originated from the works of Edelman and Tononi (2000) and Tononi (2004).

3.2.3 Metastability

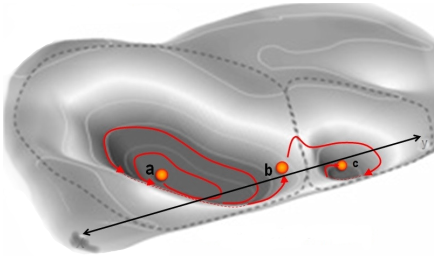
The concept of metastability is mostly used to describe a far from equilibrium complex system in terms of its movement in a stable state-space[4]. Such a system has a

open-ended intelligence and theory of assemblages. This is not to say that these concepts are not related, but the precise treatment of this aspect is outside the scope of this work.

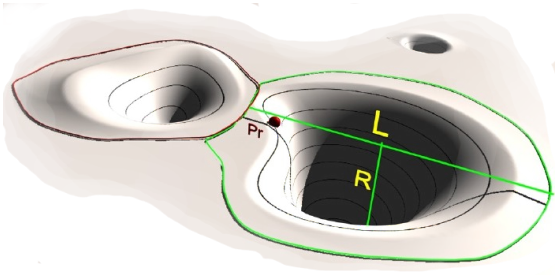
[4]The state space of a system is the set of all possible states in which the system can find itself. This is a generalisation of the intuitive concept of the concrete, three-dimensional space which an organism can explore to the abstract set of states between which a system can "move" when its properties vary (Heylighen, 2015a, p. 69). No matter how multi-dimensional, large or even infinite a state space, it is usually considered stable and invariable. Further, the existence of a clearly defined fitness function and a measure of a system's energy are often assumed. The stable state space and fitness function together define what is called *fitness*, *stability* or *energy landscape*.

landscape with many attractors while most of the time "stuck" in "shallow" attractors which may or may not represent the system's state of least energy. A metastable system can be easily perturbed, in which case it moves over a border of one basin of attraction to another (Figure 3.7a). How easy or difficult it is to perturb a system depends on the shape of its energy landscape and the precise configuration of the system's parameters at the moment of perturbation. For example, if the energy landscape contains deep attractors and a system has found a bottom of one of them, it would be comparatively difficult to perturb it. On the other hand, if the energy landscape is shallow or a system finds itself on the border of a basin, a minuscule perturbation could change its whole dynamics.

Walker et al. (2004) have proposed a collection of measures for describing a system's resilience in terms of the shape of its stability landscape (i.e. stable energy landscape) and immediate configuration which also apply for describing metastability. A system's state has three attributes: (1) *latitude* – the width of the current basin of attraction, determining how much system parameters have to change in order for it to "move" between basins; (2) *resistance* – the depth of the basin of attraction, determining how easy or difficult it is to change the system and (3) *precariousness* – the current state or trajectory of the system, in terms of how close it is from the border of a basin of current attractor, determining the perturbation energy needed for a system to cross the border and change the dynamics (Figure 3.7b).



(A) **"Classical" metastability:** A metastable state of a weaker bond (a), a transitional 'saddle' configuration (b) and a stable state of a stronger bond (c). Adapted from Harrison (2013)



(B) **"Precariousness":** Three-dimensional stability landscape with two basins of attraction showing, in one basin, the current position of the system and three aspects of resilience, L = latitude, R = resistance, Pr = precariousness. Adapted from Leuteritz and Ekbia (2008).

FIGURE 3.7: Meta-stability described in terms of attributes of stability landscape.

Weinbaum and Veitas (2017a) propose an "extended" concept of metastability which, apart from the aspects discussed above, has a *fluid fitness landscape*. Recall that a stable fitness landscape is defined by (a) a state space – relations among all possible states – and (b) a fitness (or goal) function. A "classical" concept of metastability assumes that these two parameters are fixed as they define the identity of the system itself. The "extended" metastability offers the possibility of describing the dynamics of a system with *fluid identity* where fluid state space is influenced by movements and interactions of its lower scale systems operating on it rather than defined *a priori*. An example of a "classically" metastable system is water at $0\,^{\circ}\mathrm{C}$ temperature. If the water is still, it stays in a liquid state (even below the temperature of $0\,^{\circ}\mathrm{C}$), but if it is perturbed by vibration, it collapses into the state of ice. In the framework described above, stable physical properties of water molecules define the state space at $0\,^{\circ}\mathrm{C}$ temperature – the precarious configuration of the system on the borderline between basins of attraction defining the liquid and solid states. A system that is best approached by the concept of "extended" metastability is that of

financial markets, which are an example of a sociotechnological system. Participants of financial markets choose their actions on the basis of relative prices of stocks and their movements. The price (and its movements) of a stock is determined by expectations of participants in the financial markets (and buy/sell actions that these expectations get resolved to). Therefore participants heavily influence the system's "state-space", which they try to predict and act upon. Stock market crashes and bank runs caused by no more than panic of a critical mass of participants of a system, as well as inflated price bubbles, are examples of operation of "extended" metastability. More generally, the fitness landscape of such a system is the product of a form of social agreement (implicit or explicit) of participants of a system rather than defined exogenously.

A concept closely related to "extended" metastability is reflexitivy. *Reflexivity* refers to the circular relationships between cause and effect when each element both affects and is affected by other elements. In particular, it refers to a feedback relationship between observer (i.e. intelligent agent) and observed (i.e. environment): any examination and action of agents "bend" the environment and affect the perception and further decisions by the same agents. From its very definition, the sociotechnological system is a reflexive system with a vast number of feedback loops. Reflexivity, blurring the distinction between causes and effects, makes systems difficult to analyse and predict. The contribution of reflexivity to the dynamic properties (e.g. fragility – see Section 2.2.4) of a system depends on the kind of feedback mechanisms that operate. A negative feedback has a stabilizing effect on the system's behaviour as it resists any change in the state of the system. This is not the case with positive feedback, which has the opposite effect of destabilizing the system by amplifying any disturbance. The crucial aspect of the reflexivity property for a sociotechnological system is that patterns of modelling and representation of the world have a decisive effect on the type of feedback loops which develop in it. This is grasped by the example of "extended" metastability when for example a stock market crash is caused by a positive feedback: a price of a stock randomly fluctuates down which may bring stressed traders to sell that stock because they predict a further decrease. Which indeed becomes a self-fulfilling prophecy: each sale order further reduces the price and drives an avalanche of sale orders which may eventually crash the stock market.

Walker et al. (2004) formulate the concept of *panarchy* to account for the reflexive nature of a complex metastable system in terms of its resilience. Panarchy considers how latitude, resistance and precariousness attributes are influenced by the scales above and below the focal scale of a complex system with scalable structure (see Figure 3.8 on page 73). Further, Marsh and Onof (2008) try to provide a more formal account for relations among scales in their model of stigmergic social epistemology applied to distributed cognition. They propose a modified particle swarm algorithm where the global fitness function f_t is defined as dependent on the actual configuration of particles at time t within the state space:

> [..] this would amount to having the moves made by the individuals (either each individual's trajectory, or perhaps only the evolution of the group's best position) have an impact upon the actual shape of the landscape – one could imagine these individuals' movements causing earthquakes or landslides, for instance (ibid., p. 11).

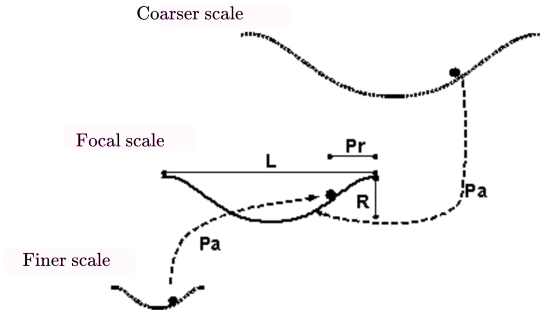
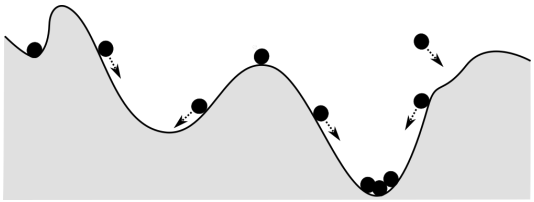


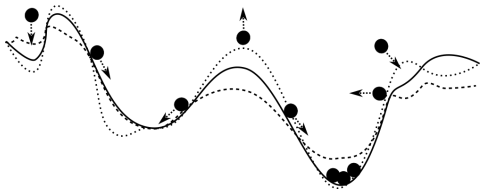
FIGURE 3.8: **"Panarchy"**: illustration of the influence of states of a system at scales above and below the focal scale, emphasizing functional aspect of the meta-stability landscape and the source of its fluidity. Adapted from Walker et al. (2004).

Weinbaum (2013) and Weinbaum and Veitas (2017a) develop an abstract model of synthetic cognitive development based on scalable cognition and scalable individuation which conceptually describes how recursively nested scales or stratas of assemblages emerge in the process of individuation and boundary formation. The model is an integral part of the philosophical framework of open-ended intelligence and will be discussed in Section 3.3.3.

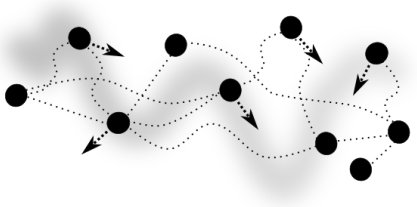
In summary, (1) the "classical" metastability is the phenomenon when a system is permanently in a configuration other than the system's state of least energy while the (2) "extended" metastability describes the situation when basins of attraction are in permanent flux so that a system has no defined stable state of least energy to begin with. The extended concept of metastability allows us to establish a more formal connection between the theory of individuation and complex adaptive systems. The "order from noise" principle of complexity science (see page 38) can be intuitively grasped by imagining the state space in Figure 3.9a being "shaken" by an influx of additional noise. The energy from the noise increases the probability of a system overcoming the "saddle" configuration of local minimum and ending up in a different basin of attraction. In the case of a fluid state space, the noise does not "shake" the state-space, but rather changes its very configuration (see Figure 3.9b).



(A) Stability landscape.



(B) Metastability landscape.



(C) Reflexive metastability: patterns of interaction among particles can be seen as a fluid network structure which, in case of persistence, gives rise to metastability landscape.

FIGURE 3.9: Degrees of metastability.

An important implication of this difference is that in a fluid state space we can relate the transformations of a state space configuration to the movement of a system in it, without positing an external source of noise or energy, as is usually done within the framework of classical metastability. Furthermore, an extreme case of a fluid state space would be the situation where there is no observable energy landscape to begin with, apart from more or less fuzzy patterns of system's behaviour (see Figure 3.9c). The relation of system's movements in a fluid state space to the

configuration of that state space is crucial for understanding progressive determination – the mechanism of individuation.

3.2.4 Progressive determination

Progressive determination is the process which describes the evolutionary development of a metastable system – how the metastability (Figure 3.9b) and then stability (Figure 3.9a) landscapes of a system are determined by the unconstrained interaction of a population of elements with no landscape to begin with (Figure 3.9c). Progressive determination is an abstract mechanism of individuation of complex systems, whereas it can be said that the more determined a system's metastability landscape, the more concrete individuality it possesses.

An extended metastable system is reflexive in that the "movement" of a system in its energy and fitness landscape changes the landscape which recursively influences – i.e. progressively determines – further movements of the system. Progressive determination can be therefore seen as a chain of transformations where an operation transforms a structure and a structure in turn transforms an operation (Weinbaum and Veitas, 2017a). More formally it is represented as follows:

$$...S_1 \rightarrow O_1 \rightarrow S_2 \rightarrow O_2 \rightarrow S_3 \rightarrow ... \rightarrow O_n \rightarrow S_{n+1}...$$

- operation O_i is a function which transforms one structure to another: $S_2 = O_1(S_1)$;
- likewise, structure S_i is a function which transforms one operation to another: $O_2 = S_1(O_1)$;
- note that $S_1 \neq S_2$ and $O_1 \neq O_2$ – they are *different* functions;
- the symbol $\rightarrow$ denotes the *relations of dependency between* the transformations, so that every transformation depends on the full history of previous transformations.[5]

The concept of progressive determination embraces the philosophy of information of Simondon in that information that guides reflexive processes in a metastable system individuates within the process itself and is not in any way *a priori* defined (see Section 3.2.1). In order to become coordinated, individual processes in a diverse population first have to find and negotiate the basis for their interactions and select "meaningful" ones while developing criteria of meaningfulness at the same time.

Stigmergy – a somewhat better known and researched concept in biology and computer science (see page 47) – is a special case of progressive determination in that it is an indirect, mediated mechanism of coordination between actions. In stigmergy the trace of an action (i.e. *operation*) left on a medium (i.e. *structure*) stimulates the performance of a subsequent action (Heylighen, 2016). Stigmergy relies on the cybernetic relation of agent-environment-agent-environment through ongoing and mutual modification or conditioning. Marsh and Onof (2008) note that *emergence* – a novel behaviour arising from the lower scale of a system – and *immergence* – an

[5] I.e. it should *not* be understood as a piping of inputs and outputs through the chain of immutable transformations.

individual action informed by the global state / higher scale of a system – go hand in hand and should be approached as "perpetual iterative looping" in a stigmergic way.

Now, after establishing relations between individuation of identities, metastability of systems made of networks of interacting elements and emergence as a result of progressive determination of structure and function, it is the place to introduce an important aspect of the computational approach which will be the basis of the rest of this work – starting from Chapter 4. Our proposal is to represent the stigmergic environment and the structural aspect of progressive determination by a *graph data structure* implemented in a computational medium. Furthermore, we suggest representing the operations of progressive determination and stigmergic actions as computational processes reading from and writing to this graph data structure. Such a framework could be said to implement a *computational stigmergy* or *stigmergic computing*. Next we turn to discussing how the mechanisms of individuation, progressive determination and metastability reveal themselves in the context of becoming intelligent – i.e. individuation of cognition.

3.3 Individuation of cognition

Open-ended intelligence is manifested in concrete environments and embodiments via the process of progressive determination described in previous sections. Recall the continuum of freedom and constraint where open-ended intelligence represents the unconstrained potentialities of different behaviours while a deterministic algorithm represents the most constrained algorithmic behaviour. The freedom and constraint continuum relates open-ended intelligence to "general" and "narrow" AI, human-level intelligence, and deterministic algorithms by conceiving a mechanism of progressive determination of constraints through which more observable manifestations of intelligence in specific situations and environments could emerge. This is what we refer to as *individuation of cognition*. Figure 3.10 shows how individuation of cognition relates to the philosophy of individuation and concepts discussed above, including degrees of metastability and the process of "concretization", which are different perspectives towards the same process.

3.3.1 Pre-, fluid and fully formed individuals

We necessarily think of intelligence in terms of its already consolidated and manifested forms in concrete embodiments – humans, animals, fish, robots, fungi, societies... Yet from the perspective of open-ended intelligence, the very process of becoming intelligent – the process of phylogenetic, ontogenetic and cognitive development – is much more primary and important for understanding intelligence and mind than any of its concrete and observable instantiations. Nevertheless, the process *can* be looked at in terms of the structures it gives rise to. We usually refer to such cognitive structures and, moreover, their embodiments as *individuals* or *agents*.

Looking at intelligence from the perspective of the processes of its emergence rather than at already formed intelligent agents, allows us to see beyond the boundaries of nature versus nurture and environment versus organism. Actually, the difference between "nature" and "nurture" can be observed only from the perspective of a single point in time marking the specific stage of developmental process. Yet

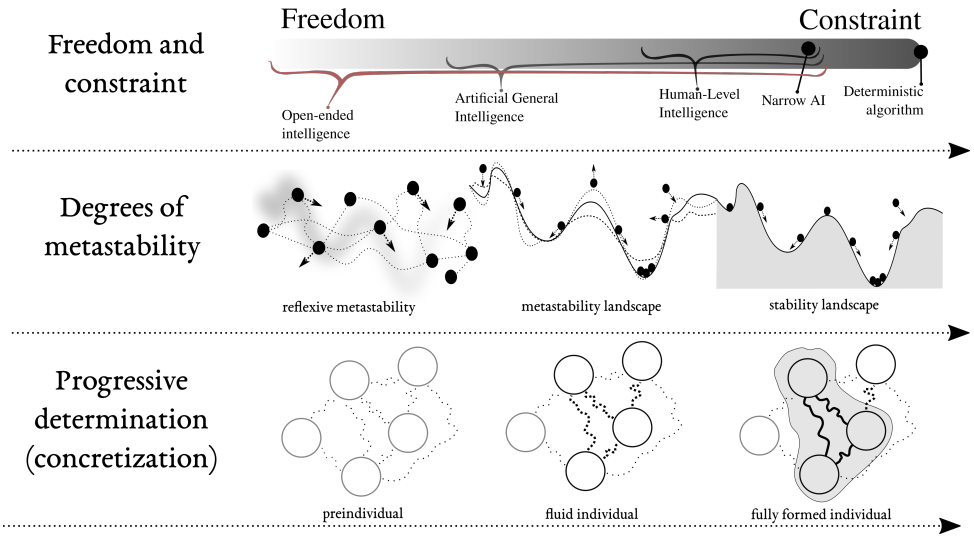


FIGURE 3.10: Individuation of cognition from the perspectives of progressive determination of constraints (Figure 2.1), stabilization of a landscape (Figure 3.9) and "concretization" (Figure 3.4).

in terms of the evolutionary process at large, the distinction itself does not make much sense – obviously phylogenetic processes are influenced by the environment at least in terms of selective pressures in the process of natural selection. This perspective makes the distinction between agent and environment much less clear than is usually posited both in evolutionary biology and artificial intelligence research. Furthermore, it is said that the evolution of intelligence internalizes certain aspects of environment to the organism thereby allowing it to adapt (see the concept of vicarious selector in Section 2.2.1). Yet at the imaginary start of the process, when nothing was yet "internalized", what was the organism that could have started the very process of internalization?

In the process of individuation, individuals are not preceded by already individuated entities or principles that instruct the trajectory of their formation, but by a state of affairs which is as yet undetermined – the *preindividual*. Even after an individual has reached a relatively stable state, the preindividual is not exhausted and persists in the individual. This is what allows its subsequent individuation or becoming. The unity characteristic of fully individuated beings (i.e. identities) which warrants the application of the principle of the excluded middle, cannot be applied to the preindividual (Weinbaum, 2015). *Fluid individuals* do not have clearly consolidated boundaries or agency and in that sense could be understood as "not yet fully formed". On the other hand it would not be correct to think of fluid individuals and identities only as intermediate states in the process of formation of individuals. Quite on the contrary, it makes sense to perceive more or less fluid individuality as a more befitting expression of intelligence than fully formed individuals.

A study and debate around plant intelligence (Firn, 2004; Trewavas, 2003, 2004) provides fertile insights into general intelligence, spanning all its possible manifestations. Notably, it reminds a broad definition of intelligence by (Stenhouse, 1974) as “adaptively variable behaviour within the lifetime of an individual”. The broad definition does not relate intelligence to its functional expressions, such as movement, and implementing mechanisms, such as nervous systems or brains. Intelligence is a property that can be attributed to living organisms of all kinds and therefore is

inseparable from life. Every form of life is intelligent to an extent that allows the organism to "adaptively vary".

3.3.2 Scales of individuation

Simondon emphasizes that relations between individuals also undergo individuation: "A relation does not spring up between two terms that are already separate individuals, rather, it is an aspect of the *internal resonance of a system of individuation*. It forms a part of a wider system" (Simondon, 2009, p. 8). In particular, individuation never brings to light an individual in a vacuum but rather an individual-milieu dyad that defines the boundary between an individual and its environment (see Section 3.2.2). This dyad contains both a system of distinctions and a system of relations. The individual and its milieu reciprocally determine each other while developing as an integrated system wider than the individual (Weinbaum and Veitas, 2017a).

Notions of individual-mileau dyad, boundary and environment bring forth the image of scales and relations among them. An individual is identified and understood first and foremost as a part of a larger whole, which is the environment. Yet the individual itself is comprised of smaller scale components and an ecosystem of internal boundaries. The concept of different interacting scales is inescapable in evolutionary developmental thinking and has already been discussed in terms of units, levels of selection and interactions between them (see Section 2.2.2), hierarchical evolution (see page 20), meta-system transition (see page 24) and control system hierarchy within perceptual control theory (see page 43) as well as the panarchy model in the metastability context (see Figure 3.8). The model of scales of individuation is a conceptual umbrella, providing a philosophical ground for these notions in the context of open-ended intelligence.

In this model (see Figure 3.11), individuation is a process that takes place at multiple scales, both structural and functional, of the individuating system. We describe the model at some scale S, where we observe a population of agents P_s. Every agent in P_s is a product of self-organization into assemblages of simpler agents at the lower scale $S-1$. Similarly, super-agents A_s^i that emerge at scale S are the elements at the higher scale $S+1$. The individuation of agents, therefore, is taking place simultaneously at multiple scales. In most cases, lower scale agents must have more stable properties than higher scale agents. Instability of agents at lower scales would make higher level organization much less probable – see the probabilistic consideration in terms of two watchmakers by Simon (1962) (see page 20 for a classical example).

Scales differ not only structurally but also temporally. As a cognitive system individuates, complex objects emerge and their interactions may become slower in comparison to their lower scale components. Generally, the relative frequency of interactions at scale S is lower than those at scales lower than S and faster than the frequency of interactions at scales higher than S.

Following Simondon's understanding of information (see Section 3.2.1), as new individuals A_s^i emerge at scale S, new information is being created. This information is expressed in the structural and functional distinctions that become apparent at that scale. Whatever remains incompatible among the agents of the lower scale does not get expressed in the emergent new structures. Across multiple scales of individuation, these incompatibilities remain as the preindividual.

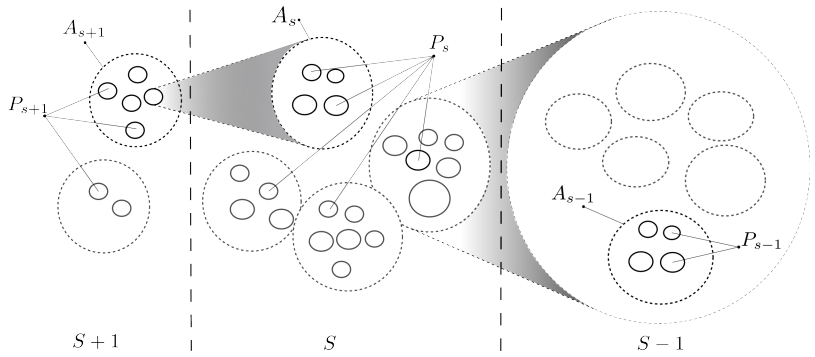


FIGURE 3.11: Relationship among scales, populations and boundaries. The focal scale of analysis is S. $S+1$ is the higher scale while $S-1$ is the lower scale. P_s denotes a population of agents at scale S. Solid circles denote the agents of population P at any scale. Dashed lined circles denote super-agents at any scale – e.g. A_s at the centre of the figure denotes a super-agent that emerges from the interactions of agents in P_s. Super-agents at scale S are the agents of the population P_{s+1}. The i^{th} super-agent at scale S is denoted A_s^i, the superscript is omitted if not necessary. Also, the subscript s is omitted from A or P in the text if it is redundant. Adapted from Weinbaum and Veitas (2017a).

Scales of individuation provide a conceptual tool for a holistic perspective which is often neglected when taking an analytic approach of looking at each scale separately. First and foremost, importantly for this work and specifically for the artificial general intelligence programme, the model of scales of individuation relaxes the premise of the goal-directed nature of intelligence. In the words of Powers, Clark, and Farland (1960), "higher-order perceptions are kept in their goal-states by specifying lower-order goal-perceptions; the higher-order system decides on a goal-perception for the lower-order system, but does not actually do anything to achieve it. Thus each goal-seeking system is autonomous to the extent that it must contain the circuitry for making its own feedback signal approach its given reference-level and for recording its own store of potential reference-signals for later use: but each goal-seeking system is controlled to the extent that it does not choose which of its past experiences are to serve as goal-perceptions" (ibid., p. 310). When seeing the individual-milieu dyad as a system (i.e. considering environment as part of the system), it is no longer possible simply to assume that a system is controlled by externally defined goals, as some of them are defined "within" the system. Even when it makes sense to say that lower level goals or reference values *can* be derived from the higher order goals, this derivation is not straightforward nor linear in a sense that the lower level intelligent systems actually have a say when adjusting reference values of themselves.

3.3.3 Synthetic cognitive development

Synthetic cognitive development is a model of individuation of cognition from the vantage point of an individuating agent and is a generalization of human cognitive development. Since synthetic cognitive development operates within the framework of scales of individuation, it can be viewed from internal and external perspectives (Figure 3.12). In a scalable system, every subsystem can be positioned at a *focal* scale s between higher $s+1$ and lower $s-1$ scales . A lower scale consists of a

population of elements which integrate to a subsystem at a focal scale; a higher scale consists of a population of subsystems of the focal scale.

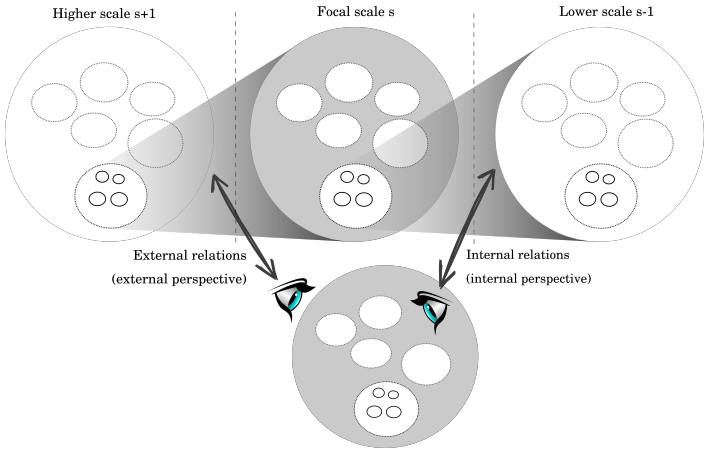


FIGURE 3.12: Internal and external perspectives to relations between scales. Adapted from Veitas and Weinbaum (2017).

Internal perspective

An internal perspective approaches the process of coordination "as if from inside" the individuating assemblage in a population of agents. As Weinbaum and Veitas (2017a) point out, the mechanisms that are responsible for bringing forth coordinated activities arise primarily from agents' intrinsic capabilities to affect and be affected by each other. The specific characteristics of the interactions, e.g. their frequency, their synchronization and coherence, have a critical influence on the way agents are connected. Such influence finds its expression in the reinforcement or suppression of connections among agents and consequently on how strongly they may actually affect each other. This is how the activity of agents within population P progressively determines the topological organization of the network of agents in it. The structural organization, in turn, affects the overall function of the individual agents by selecting interactions. Higher order agents emerge as assemblages when many incompatible elements interact and achieve a certain degree of mutual or collective compatibility. Connections between compatible agents are reinforced while other connections tend to be suppressed. In the course of such recursive selective interactions, groups of compatible agents assemble into distinct compound organizations resulting in individuated super-agents. Weinbaum and Veitas (ibid.) further clarify the criteria for compatibility utilized in the selective process and the mechanism of reflexive mutual selection taking place among agents.

Criteria for compatibility

Agents overcome their initial incompatibility by constraining each others' regimen of behaviours. In other words, there is a process of reflexive selection going on where every agent selects with which other agents in the population it can interact. Three understandings of compatibility from the simpler to more complex are presented below:

Synchronization – Agents that produce effects (become active) at the same time will tend to reinforce their connections. The kind of compatibility that is selected by this criterion is temporal coincidence, which may indicate with some probability that the synchronized agents are causally affected by either the same event or by events that are causally connected, or events that are otherwise correlated. The formation of synchronized clusters of agents is the simplest form of individuation. Synchronized groups will tend to reinforce their synchronized behaviours and suppress their out-of-sync behaviours. Examples of individuation following this criterion can be found in neural networks. The Hebbian rule that neurons that fire together also wire together is one application of this criterion. A more complex application is provided by Edelman and Tononi (2000) who hypothesize that spontaneous synchronization among groups of neurons is the basis of consciousness[6]. Both are examples of cognitive development at the scale of groups of neurons.

Coherence – Agents that produce effects (become active) in response to informative patterns (not necessarily synchronized) that represent the same category or type, or a group of mutually supporting logical propositions, or a group of associative patterns, will tend to reinforce their connections. The kind of compatibility that is selected by this criterion is much more abstract than synchronization and requires a context of operation. The agents connecting according to this criterion form coherent clusters. Clearly, in this general form, the coherence criterion is underspecified. Coherence will normally operate as a selective criterion only in populations of relatively complex agents where the information that agents exchange already signifies lower level individuated objects. Such objects provide the context that further determines what coherence means. Thagard (2002) explains coherency as the joint property of propositions that tend to be selected together or rejected together when tested in the context of a certain domain or state of affairs (see also Section 2.3.4). Thagard's understanding of coherence distills a second kind of compatibility, which we can generally describe as compatibility in signification or meaning.

Coordination – Coordination is broadly defined as functional compatibility. In fact, synchronization and coherence can be described as special cases of coordination. Agents that interact, process information and produce effects that jointly realize a function or a goal are said to coordinate their operations, thus presenting functional compatibility. Connections among agents that support the coordinated activities will be reinforced while those that disturb the coordinated activities will be suppressed. The agents connecting according to this criterion will form coordinated clusters. As in coherence, coordination will operate as a selective criterion only in populations of relatively complex agents and where the information that agents exchange already signifies lower level individuated objects and their relations. Such objects provide the context that further determines the nature of the function or goal that is performed by the coordinated clusters. Autopoiesis (Maturana and Varela, 1980) is an illustrative example of a self-organized coordination. Remarkably, autopoeisis is a function that operates in relation to the same cluster of agents that realizes it and therefore does not require an outside observer for its definition. Yet functional compatibility is not limited to this family of self-determined functions. Coordinated clusters may also emerge in response to signals mediated by the

[6] See also Tononi, Sporns, and Edelman (1992) for a more detailed description.

higher scale and as such are external to the population of agents under consideration. These signals guide selection by providing an external criterion of functional efficacy. In other words, the actual compatibility criterion of coordination may be either self-produced or external. Emergent agents, accordingly, may be self-coordinating or coordinated in relation to an external state of affairs, as well as be affected by both modes. For an overview of coordination mechanisms see Heylighen (2013).

Reflexive mutual selection

Individuated entities are the product of a recursive resolution of incompatibilities via synchronization, coherence and coordination. It is a process of reciprocal selection where agents within a population repeatedly select communication links and interactions that increase compatibility according to these criteria. The reinforcement of compatible interactions and suppression of incompatible interactions progressively determine assemblages of integrated agents within the population. Structural changes in the network of agents drive further selections and this progressive determination, which continues until the network achieves relative stability with consolidated super-agents. At this elementary level, individuals emerge as products of an evolutionary developmental process: the heterogeneity of agents in population P provides the variation and the various compatibility criteria provide the selective elements of the process. The retention of compatible clusters is inherent in the process since by definition mutual compatibility among agents is preferred and reinforced. Otherwise, no individuation and no cognitive development could have taken place.

Inspired by Edelman's theory of neuronal group selection (Edelman, 1987; Edelman and Gally, 2013; Tononi, Sporns, and Edelman, 1992) the reflexive and recursive characteristics lie at the basis of the synthetic cognitive development which extends neuronal group selection to general networks of agents. The selective criteria of compatibility that are derived from the theory of individuation extend the synchronization criterion in the case of neuronal groups. Reflexive mutual selection (termed "reentry" by Edelman) is a mechanism operating within a network of interacting agents. Consider two groups of agents A and B (see Figure 3.13). Each group contains similar agents with some variety in their pattern of behaviour. The groups are interconnected internally and across. Following a signal produced by some agent in group A, a subset of agents in group B will respond by producing signals too. This activation will spread both internally in B and across back to A (where some of the agents are already active too). A subset of agents in A will respond to the signals coming from B such that a chain reaction of signals will propagate back and forth between A and B. In some cases, after a few cycles of exchange, a signal, whether from an agent in B or A, will be received by the initiating agent and will cause it to produce a signal similar to the one that initiated the whole exchange. If this happens, a closed activation loop begins and the groups will enter a period of sustained mutual activation that will continue until it is disrupted by other signals. Sustained activation patterns and sequences of interactions that arise in a similar manner within the population of agents are the products of what is called a reflexive mutual selection process.

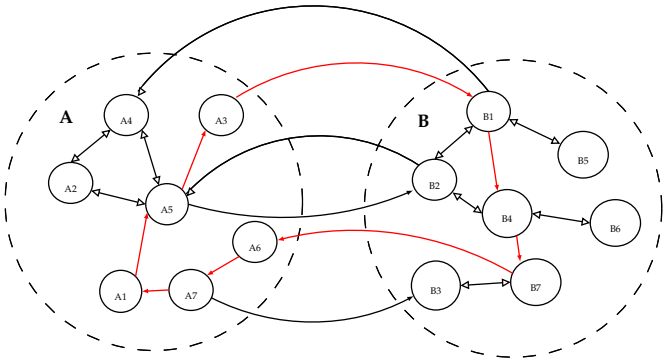


FIGURE 3.13: Two connected groups A and B and the formation of sustained mutual activation. The signal propagation path of the sustained activation is indicated in red. Other paths such as A3-B1-B2-A5-A3 are topologically possible but are not selected because activation depends also on the informational content and timing. Adapted from Weinbaum and Veitas (2017a).

External perspective

An external perspective to synthetic cognitive development approaches individuating assemblages in a population of agents by looking at them "as if from outside" (see Figure 3.12) trying to capture the emergent properties of assemblages as super-agents.

Integration and disintegration

In a dynamic population of interacting agents where the processes of territorialization and deterritorialization (see page 67) are active, we can observe emergence of super-agents with distinguishable behaviours, properties and identities – i.e. the *integration*. Likewise, we can observe change, adaptation or dissolution of these behaviours and properties over time – i.e. the dissintegration. The observation of integration and disintegration of super-agents happens on one level above the focal level, where these processes happen from the interaction among lower level elements – hence the "external" perspective.

Integration is a process which can happen locally or globally in a system and leads to the higher levels of coordination among some elements of its population at any scale. Clusters of elements which coordinate more strongly among themselves than with the rest of the population start forming an assemblage which, after reaching a certain level of internal coordination and resilience, can be identified as a newly formed subsystem with unique characteristics.

Disintegration is obviously the process in the opposite direction from integration: it leads to a lower level of coordination among elements of a given subsystem, ultimately reaching a level where the boundary between elements within the subsystem and elements outside the subsystem dissipates - i.e. it disintegrates and no unique properties can be observed at a higher level.

Despite being always present, processes of integration and disintegration are never symmetric: at every given moment either one is stronger, giving rise to the

complex dynamics of a complex adaptive system in an ecology of other complex adaptive systems. The interplay between the processes of integration and disintegration of variable strength and the importance of this interaction for the growth of the cognitive system is captured by the scheme of synthetic cognitive development (see Figure 3.14). The maintenance of the interaction of the processes of integration and disintegration in a synthetic cognitive system is instrumental for sustaining its resilience and enabling open-ended development.

The lesson of complex adaptive systems is that processes of integration (towards order) are as important for the self-organization of the system as processes of disintegration (towards fluidity).

Cognitive dissonance

The approach to cognitive development as an interaction or sequence of integration and disintegration cycles is supported by several theories. Leon Festinger's theory of cognitive dissonance, developed in the 1950s, focuses on a state of mind holding two or more elements of knowledge which are relevant but inconsistent with each other (Harmon-Jones, 2012). It is arguably the normal state of an intelligent agent engaged in a life-long activity of making sense of its environment. The theory proposes that incompatibility of the elements creates a state of discomfort or "dissonance" which is proportional to the degree of incompatibility – the lack of integration. Further Festinger hypothesized that persons experience an arousal – usually unpleasant emotions – due to the dissonance which motivates them to engage in "psychological work" to reduce the inconsistency. Cognitive dissonance theory in its original form generally enjoys experimental support. Particularly interesting are experiments showing that during the state of dissonance individuals evidence arousal and report negative affect (ibid., p. 2). Studies in cognitive neuroscience indicate the tendency of a cognitive system to choose a single explanation of sensory experience by constraining multiple possibilities, thereby reducing internal uncertainty or dissonance. For example, the entropic brain hypothesis of Carhart-Harris et al. (2014) points to the association between perception of identity and organized brain activity. The dynamic core hypothesis of Edelman and Tononi (2000) likewise connects concepts of immediate consciousness with synchronized activity of neuronal groups and areas in the neocortex.

These observations indicate the tendency of a cognitive system towards increased coherency both internally and in its relationships with the environment. Nevertheless periods of reduced coherency are necessary for the cognitive system in order to explore the possibilities of higher coherency – a *cognitive development*.

Arousal and emotion

Contrary to the established scientific opinion of the end of 20^{th} century, feelings and emotions are just as cognitive as any other percepts (Damasio, 2008, p. 16) and their role cannot be overlooked when considering the development of a cognitive system. While currently the importance of emotions and feelings for the overall operation of a cognitive system is increasingly accepted, the integration of an "emotional system" into the model of cognition is still problematic. Damasio (ibid., p. 284) proposes a view of emotions as an immense collection of changes occurring in both brain

and body, usually prompted by particular content while being felt as the conscious perception of those changes. This proposal is strikingly similar to the two-factor theory of emotion by Schachter and Singer conceptualizing emotion as general arousal plus a cognitive label attached to it (Cooper, 2007, p. 58). The state of arousal starts a chain of events within an organism which usually leads to the decrease of arousal. These events can take a form of internal "psychological work" (Harmon-Jones, 2012) or external actions in the environment, both of which can be considered as sense-making activities. Further, Damasio (2008) differentiates between *primary emotions* and *secondary emotions*. Primary emotions are "wired from birth" and constitute what is understood as drives and instincts. Secondary emotions are acquired by creating systematic connections between primary emotions and categories of objects and situations (ibid., p. 151).

Cognitive development

The theory of cognitive development (see Section 2.3.6) posits identifiable patterns of individuation of the human cognitive system which are described as developmental *stages* (Piaget, 1971) or *truces* (Kegan, 1982), usually ordered in predictable sequences. Cognitive development theories generally describe an "evolution of meaning" (ibid.) – recursive subject and object relationships in which the subject of the previous stage becomes an object during the next stage. The process is not linear, but rather is manifested through sequences of integration and disintegration of cognitive structures (i.e. developmental truces).

Human cognitive development is usually understood as a predictable and finite sequence of developmental stages. Weinbaum and Veitas (2017a) argue that both the predictability and finiteness of cognitive development are not ingrained nor necessary properties of the process but rather constitute historically shaped superficial characteristics. For example, the relative stability of observable stages of child development are related to more or less stable external influences of parents, peers and society as well as to genetic predispositions. Likewise, the fact that mature individuals rarely undergo transitions to higher levels of cognitive development is possibly related to reduced environmental pressures to engage in the "psychological work" that is needed for such transitions. The rationale of seeing cognitive development beyond its observable predictability and finiteness is instrumental for the framework of synthetic cognitive development, which aims to describe the genesis of a general cognitive agency as a continuous individuation process.

3.3.4 The scheme of cognitive development

Weinbaum and Veitas (ibid.) propose a scheme which conceptualizes cognitive development as an observable sequence of integration and disintegration processes progressively determining the cognitive complexity of an agent. The progressive nature of cognitive development is manifested by *increasing capacity of sense-making*. This process does not follow a trajectory of monotonous adaptation but rather advances in a punctuated manner going through relatively stable stages. The enactive nature of sense-making (see Sections 2.3.7 and 2.3.8) implies a reflexive relation between system and environment. At every state, both the cognitive system and the environment have more than one option to relate to each other. Therefore every state of the interaction is characterized by a unique trade-off between *freedom and*

constraint (see Section 2.1.5) of the cognitive system when choosing the future development trajectory. Additionally, system–environment boundaries are themselves subject to variation. We propound that an immediate configuration of the cognitive system in terms of the *freedom and constraint* trade-off in humans is closely associated with the level of experienced cognitive dissonance, a suggestion supported by the entropic brain hypothesis of Carhart-Harris et al. (2014). The system achieves higher levels of cognitive complexity via periodic fluctuations in its level of cognitive dissonance. When the cognitive dissonance of a system is low, it undergoes constrained periods of development with more predictable developmental trajectory. When cognitive dissonance is high, the future trajectory of a system becomes more divergent. A further hypothesis is that emotions are mechanisms that guide the selection of the developmental trajectories of the cognitive system by modulating the sensitivity of the system to environmental stimuli (Weinbaum and Veitas, 2017a, p. 22).

Figure 3.14 is a scheme of cognitive development as a variation of cognitive dissonance versus coherence of a system which can be mapped to certain cycles. These cycles emerge from the attempt to balance opposing tendencies to suppress the unpredictability of the cognitive system on the one hand and keep it open for change on the other.

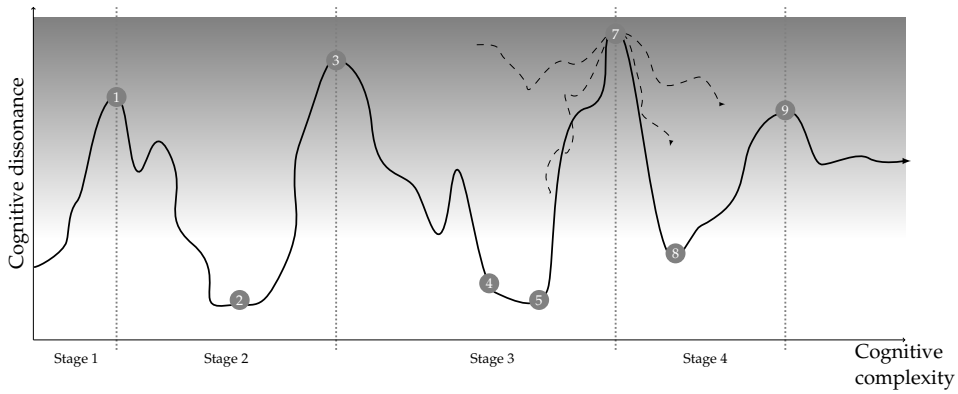


FIGURE 3.14: **A scheme of synthetic cognitive development** qualitatively visualizing the dependency of increasing cognitive complexity on the variation in the level of cognitive dissonance of a system. The bold curve represents the actual developmental trajectory. Circles with numbers represent states of development, arbitrarily chosen for illustration. States (1), (3), (7) and (9) mark high cognitive dissonance states where the system has the highest possibility of "choice" between alternative developmental trajectories. Dashed lines are drawn at stage (7) to illustrate multiple possible trajectories that are actually present at every point along the developmental trajectory. States (2), (4), (5) and (8) mark stable periods when the operation of a cognitive system is constrained. Stages 1, 2, 3 and 4 on the horizontal axis illustrate cognitive development stages as described by the developmental psychology representing punctuated manner of increase in cognitive complexity.

3.4 Summary of the chapter

This chapter introduces and discusses the conceptual model of open-ended intelligence and, on the one hand, provides a conductive and integrative perspective to established theories, modes of thinking and their problematics introduced in Chapter 2. On the other hand, the conceptual framework presented here serves as the

backbone for the computational model and architecture of open-ended decentralized computing developed through Chapters 4 and 5.

The nutshell explanation of the conceptual model is as follows. The process of individuation of intelligence happens within a population of interacting primary elements or agents when patterns of interactions among them become more or less persistent or stabilize. These patterns correspond to clusters or *assemblages* of different levels of consolidation and can be loosely distinguished as *preindividual*, *fluid individual* and *fully formed individual*, in order of increasing consolidation. The emergence of stable patterns is facilitated by the progressive process of signification of meaning of signals being exchanged and the selection of mutually meaningful communications among agents in the population. All interesting observable manifestations of real-world intelligence are fluid individuals, including humans, animals, plants and fungi, albeit manifesting different consolidation degrees. Furthermore, fluid individuals, which are partially persistent assemblages of elements, are characterized not only by structured interactions inside an assemblage, but also structured interactions across the boundaries with other members of the population. These other members together constitute the *environment* of the fluid individual. The structured interaction of the now consolidated assemblage (an agent) with its environment is explained as the movement in the metastability landscape, where the landscape is itself influenced by this movement. This is the model of interaction between scales. Moreover, since the metastability landscape itself is comprised of interacting elements, it is understood as an agent at a higher scale. All scales of assemblages and their populations recursively interacting as agents with their environments constitute the scalable model of individuation. Assemblages of agents are determined via a history-dependent process where the interactivity of elements create patterns which further influence the interactivity and in this way give rise to the mechanism of *progressive determination* – the abstract mechanism of individuation of cognition – where function determines structure and structure determines function. Importantly, the process of individuation is not directional and can lead to both integration and disintegration of patterns of communication and assemblages of agents. The scheme of *synthetic cognitive development* conceptualizes the apparent stages of human cognitive development as relatively stable periods of otherwise alternating cycles of integration and disintegration of cognitive structures and patterns of activity within a system. Notably, the notion of synthetic cognitive development is not constrained by an observed number of cognitive development stages in humans and considers cognitive development and individuation of cognition as a general divergent phenomenon.

Chapters 4 and 5 utilize this conceptual framework in order to conceive the process of individuation of cognition in computational terms and design a general software architecture for the implementation of the simulation modelling engine. Note that the conceptual model is not descriptive in the sense that it does not describe any concrete manifestation of intelligence, but rather a framework capable of expressing any form of intelligence. Likewise, the computational model and architecture developed in the following chapters is general as it conceives the computational expression of the scheme of synthetic cognitive development without limiting itself to any concrete forms of intelligence, which are provided only as examples.

Chapter 4

Decentralized computing for synthetic cognitive development

4.1 The power of computational metaphor

Chapter 1 showed how the thread that relates interdisciplinary aspects of the conceptual framework, computational model, use-cases and simulation modelling experiments in this work is the "mechanistic" perspective towards an intelligent complex system. Computational thinking is a perfectly valid and the most befitting way to actuate this perspective. Within the framework of open-ended intelligence, synthetic cognitive development as a general phenomenon arising in complex systems extends the concept of natural evolution (Weinbaum, 2018). The approach that all that evolution – the ultimate process that creates order from disorder – does is to create knowledge (see Section 2.2.1) asks for application of the best known conceptual and technical tool for reasoning and implementing processes of information signification, creation, communication and actuation – the *computational metaphor* (Kelly, 1998):

> Once nature was described as a body, then a clock in the age of clocks, then a machine in the industrial age. [...] To explain how our minds work, or how evolution advances, we apply the pattern of a very large software program processing bits of information. None of these historical metaphors is wrong; they are just incomplete. Ditto for our newest metaphor of information and computation. [.. it] can't be the most complex immaterial entity there is, just the most complex we've discovered so far. We might eventually discover that exotropy[1] involves quantum dynamics, or gravity, or even quantum gravity. But for now, information (in the sense of structure) is a better analogy than anything else we know of for understanding the nature of exotropy (Kelly, 2010, p. 64).

The abstract mechanism of individuation of cognition, which is at the core of synthetic cognitive development and open-ended intelligence, is the progressive determination of constraints (see Section 3.2.4). Approached from the perspective of the computational metaphor it is precisely the mechanism of creation and propagation of information in a decentralized system. Extending this parallel further, evolution

[1] *Exotropy*, the term first coined by Max More in 2003, denotes "an evolving framework of values and standards for continuously improving the human condition". The technical equivalent of exotropy is negative entropy.

at large can be seen as myriad fuzzily localized progressive determination processes branching to the interacting pathways of evolutionary development. These pathways in the case of biological evolution are then categorized as kingdoms, classes, phylums, orders, families, species and finally individuals.

Cognitive development of an individual intelligent being is but a small subprocess in the evolving network of life (see Section 2.2.1) yet following the same principles and mechanism – progressive determination of constraints within a population of independent, heterogeneous and interacting processes. In this chapter we formulate this process with the help of the computational metaphor in terms of *open-ended decentralized computing*.

4.2 Through the lens of computation

The artificial intelligence programme has gained much of its creative momentum over the years from the tensions between understanding and modelling intelligent behaviour in terms of the number of interdisciplinary ideas, including (1) *sub-symbolic* versus *symbolic* representation and processing, (2) *deterministic* versus *non-deterministic* computation, (3) symbol grounding, (4) notions of *selective* versus *descriptive* information and more. Each of these paradigms consists of a complex set of assumptions, premises and techniques which are often shared up to the point where the difference between paradigms themselves becomes fuzzy. Moreover, thinking in terms of a single paradigm is not sufficient for covering all (or even most) interesting aspects of machine intelligence as well as its "natural" counterpart. We discuss here these paradigms briefly in the light of computational metaphor and its relation to open-ended intelligence.

4.2.1 Symbolic versus sub-symbolic

The idea of symbolic processing rests on the premise that all computation is a syntactic manipulation of symbols, and therefore any behaviour can be described and modelled by (a) defining a correct set of symbols (i.e. entities) (b) defining correct rules of symbol manipulation (i.e processes). An implicit and sometimes taken for granted assumption within the symbolic paradigm is that symbols and rules are singularly related to clearly defined meanings which are known to be true to humans (programmers, philosophers, mathematicians) and therefore can and should be translated into symbolic systems in order for them to exhibit intelligent behaviour.

A way to see how sub-symbolic differs from symbolic is to try to see what processes **do not** fit under the above definition. In terms of the first part of the definition regarding symbols and rules of manipulation, the sub-symbolic paradigm complies as well, especially considering our computational perspective: since it can be implemented in a computational medium it is surely a manipulation of symbols, albeit something that does not make much sense to humans – zeros and ones. So the difference seems to be not in whether manipulation of symbols is taking place via the rules, but rather what kind of symbols, what kind of rules and, most importantly, how their so called "meaning" is being encoded and manifested. It is obvious that for moving forward we will have to discuss the concept of a symbol and how symbols acquire "meaning" by being grounded in "reality" – the *symbol grounding problem* of artificial intelligence (Harnad, 1990).

A good start is Luc Steels' observation that the concept of symbol is used in two distinct ways: as a *c-symbol* of computer science and *m-symbol* of meaning-oriented symbols in the tradition of the arts, humanities, social and cognitive sciences (Steels, 2008, p. 8). Many of the perceived and debated problematics concerning differences between symbolic and sub-symbolic representations and processing seems to be related to confusion between these two usages of the concept. The formulation of a similar distinction by Carl Gustav Jung is helpful in shedding some light: a *sign* is a reference to something known; a *symbol* is a figure by which allusion is made to an unknown (Campbell, 1958). C-symbols of science, symbolic logic and computer science are *signs*. The takeaway is that signs are always unambiguously defined in a context-free manner and, consequently, the rules of the sign manipulation ("symbolic processing") are always well defined within a given theory, system or programming language. On the other hand, symbols may mean many things depending on the context of their usage and appearance. The rules of such symbol manipulation are not really defined as rules, since relations between symbols are mostly associative and highly context-dependent. In short, signs and rules are (or at least can be) unambiguously defined and correspond to being so called "symbolic", whereas symbols and associations are ambiguous by nature and are so called "sub-symbolic". It is surprising how labelling and categorization can sometimes make things more obscure.

The fundamental unsolved problem within the programme of artificial intelligence is how symbolic and sub-symbolic paradigms can coalesce. It has been demonstrated that the assumption of clear correspondence between sub-symbolic and symbolic representations does not hold. Interaction among symbolic and sub-symbolic representations and processing in intelligent systems is intricate and fluid (Heylighen and Gontier, 2019), much more than could be expressed via a strict system of rules. Understanding and modelling them should necessary involve the aspect of emergence and "meaningful" signification of information (see Section 3.2.1). Implementation of such emergence has to be supported by certain forms of embodiment and their interactions – an avenue that has been pursued by Steels (2015).

4.2.2 'Selective' and 'descriptive' information

The Jungian distinction of sign as denoting something known and symbol as denoting something unknown provides yet another entry point for asking "what is information?" in the light of the Simondonian extension of Shannon's mathematical theory of communication. MacKay (1969) propounds that two kinds of information can be distinguished on the basis of how information content is measured or at least grasped – selective information-content and descriptive information-content.

Selective information-content is used in mostly technical situations where information can be defined in terms of the answer to a question – and hence reduce the uncertainty expressed by that question. Precisely this kind of information is used in Shannon's mathematical communication theory. Selective information-content is that which allows us "to make a selection from a set of possibilities or to narrow the range of possibilities about which we are ignorant. [...] The selective information-content of a message, or of the result of a scientific experiment, for example, has to do with the number of independent choices between two possibilities which it enables us to make – the number of independent yes's or no's to which it is equivalent" MacKay (ibid., p. 11). More generally, selective information-content is related to the

statistical concept of probability – the probability of selecting a value from a known distribution (Carnap, 1955).

Descriptive information-content refers to a different situation, in which information is not related to selecting an answer from the list of possibilities, but to *describing* an unknown situation in the first place. "The apparatus which gives us most descriptive information, in this new sense of the term, will be that which yields the largest number of bricks (metaphorically speaking) for our symbolic picture, or which enables us to show the maximum amount of fine structure in it" MacKay (1969, p. 11). The relation between selective and descriptive is somewhat similar to the relation between selecting an item from an existing probability distribution and *constructing* the probability distribution in the first place. The former is "closed-ended" because it reduces possibilities and uncertainty, the latter is "open-ended" because it explores possibilities of descriptions and structures. Descriptive information-content is related to the concept of *inductive probability*, which refers to the probability of a hypothesis with respect to the body of knowledge given the additional body of evidence (Carnap, 1955).

An intelligent agent devising a model of complex environment necessarily employs a combination of processes utilizing both selective and descriptive information-content. Furthermore, due to limited resources of time and memory, operation in a real-world environment presents situations which require relevant information to be selected from an effectively unknown distribution – as explained by the *selection for relevance* aspect of model building (see page 29).

4.2.3 Deterministic versus non-deterministic computation

The tension between symbolic and sub-symbolic representation and processing paradigms has been well manifested through the sixty-year-old history of artificial intelligence. Not so with relation to deterministic and non-deterministic behaviour and computation, at least in a direct manner. Both terms are well defined: *determinism* is expressed in terms of an abstract process which given the same input produces the same output no matter how many times repeated, while *non-determinism* may produce any output or none at all. Rational, scientific and everyday thinking is heavily biased towards determinism, which allows one to reason about causes, effects, goals and how to reach them. However, deterministic and non-deterministic behaviour are only extremes of a continuum of probabilistic behaviours of complex systems that can be expressed by different shapes of probability distribution of outputs given the same input (see Figure 4.1). These behaviours may involve many causes of the same phenomenon, different phenomena caused by a single cause and many interacting causes leading to many phenomena – see the distinction between "preformationist", "interactionist" and "constructivist interactionst" perspectives in Figure 3.1 . The prevailing bias towards deterministic, asymptotic and goal-directed thinking introduces a powerful filter which selects only a small sub-set of behaviours, phenomena, computational processes and descriptions from the environment and the world.

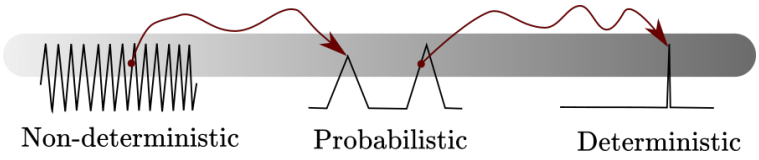


FIGURE 4.1: The continuum of probabilistic behaviours with deterministic and non-deterministic as extremes. Stylized probability distributions illustrate probabilities of an observation given the same input into the process representing a behaviour. Note that the illustration is provided for didactic purposes without claim for accuracy.

Informally, non-determinism, which is the left-most extreme of the continuum, can be understood as a uniform probability to receive any output from the behavioural or computational process, while determinism, which is the right-most extreme, a single well defined output. Probabilistic behaviour is then anything that can be observed between these two extremes. What is important is that the continuum as well as its extremes are not symmetric – something that gets obscured by the fact that non-determinism linguistically is the negation of determinism. Yet if non-deterministic behaviour can produce any output, it also can produce outputs that are explainable by deterministic behaviour. Therefore, *deterministic behaviour is a special case of non-deterministic behaviour and, consequently, non-determinism is more fundamental* than its "positive" counterpart, as illustrated by arrows in Figure 4.1.

Formal understanding of the space of behaviours and their properties within this continuum is studied by the theory of computability, which tirelessly ponders the question "what is computable *in principle*?" (Immerman, 2016). Computational complexity theory[2] has evolved from computability theory by restricting the notion of computable to computable *efficiently* (Kolokolova, 2017) within bounded time and space. Computability properties of processes are estimated by classifying them into computational complexity classes which correspond to known abstract models of computation, and many of them can be characterized by an appropriately restricted Turing machine (Vitanyi, 2009). This has positioned the Turing (1937) model of computation at the core of computational complexity theory and, consequently, the core of computational thinking and metaphor.

Not surprisingly, the Turing machine and Turing completeness are often considered concepts that define computation itself in a sense that any process that could be considered an "effective computation" is expressible in terms of a Turing machine (Stannett, 2004). The computational metaphor, penetrating and greatly influencing scientific and naive thinking since the first half of the 20th century, got shaped into *pancomputationism* – a perspective that all processes (in physics, evolution, biology, life and cognition) **are** algorithmic (Piccinini, 2015). This perspective itself became simultaneously a locus of a heated debate and a point of departure for different theories and ways of thinking. In the computer science community the debate revolves around the questions "what is computation?", "what is effectively computable?", "what is physically computable?", alternative interpretations of the Church-Turing thesis, the possibility and construction of formal and physical computational models that are more powerful or expressive than Turing's model (Nayebi, 2014). In the areas beyond theoretical computer science (notably in cognitive science, philosophy of mind and AI, complexity science) the debate influences perceptions of power and

[2]Not to be confused with complexity science of Section 2.2.4

limits of the computational metaphor for understanding complex and chaotic systems, natural phenomena and intelligence[3].

There is also a deeper level beyond this debate, very much related to the emphasis and desire of predictability of the world, thinking in terms of asymptotic processes as well as the desire to contain them in a goal-oriented manner. Goal-oriented and deterministic thinking both empowers and constrains. It would not be a great overstatement to say that it has enabled the realization of most of the technological advancements and creations of human civilization in terms of tools' and systems' engineering. Yet its constraints come to the surface when this thinking is applied to understanding, modelling of or interacting with complex adaptive systems which are by definition chaotic and unpredictable (see Section 2.2.4). Interestingly, the association of the computational metaphor and determinism may in part be an "unfortunate consequence" (very much like the one of Shannon's mathematical communication theory discussed in Section 3.2.1) of enormous impact of the early work of Turing (1937), overshadowing the later ideas of Turing (1948) which were not published for over 20 years due to being considered not suitable for publication by contemporaries (Eberbach, Goldin, and Wegner, 2004).

The classical notion of Turing computation is tightly related to the notion of decidability and the *halting problem*. Actually, the paper that established the concept of the Turing machine (Turing, 1937) specifically addressed Hilbert's *Entscheidungsproblem* – along with Church (1936a,b) and Gödel (1931). This problem required "to devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers". The Turing (1937) machine is a mechanical device that takes as input the calculable function together with its parameters and tries to solve it. If the function is solved, then the machine *halts* and outputs the solution; otherwise it works forever. Turing showed that there is no procedure that can discriminate between functions that are unsolvable and the ones that take an exponential amount of time to compute and that the *Entscheidungsproblem* is not solvable in principle – without running the actual computation. Observe that the very formulation of the halting problem defines determinism and its relation to finite resources of time and space – the issues of efficiency of computation. But the somewhat prevailing perspective that computation is *only* the process that eventually produces results (i.e. halts) and is the *only* useful computation, relates more to the "desire for predictability of the world" than any formal results of computer science or logic.

Models of computation that can compute more than the universal Turing machine of 1937 started to be explored not long after, including, e.g. oracle machines (Turing, 1938) and unorganized machines (Turing, 1948). This line of research was revived mostly at the end of the 20th century under the umbrella of *hypercomputation* or the *super-Turing computation* movement within the computer science community (Copeland and Proudfoot, 1999; Nayebi, 2014). The concept of hypercomputation refers to the "possibility that a physical or conceptual machine might be able to perform non-recursive computations, therefore stepping outside limits on computability suggested by the Church-Turing thesis" (Stannett, 2004). Besides producing many alternative computational models, the intensified research of the limits of computation first and foremost contributed to crystallization and better understanding of the question "what is computation".

[3]Francis Heylighen (2016), private conversations

Piccinini (2011) observes that the original Church-Turing thesis involves the intuitive description of "efficient calculability" and offers a notion of *usability constraint* as well as a refined interpretation of the thesis. This interpretation is useful for grasping borderlines between conceptual, mathematical, physically realizable notions and models of computation as well as their classical and "hyper" aspects. Most importantly, it provides a stepping stone for describing the computational model of progressive determination, which is the goal of this chapter.

4.2.4 Closed computing

Piccinini (ibid.) provides a revised interpretation of intuitive aspects of the Church-Turing thesis by first distinguishing its **mathematical** aspect, which is the thesis supported by the original arguments, and **physical** aspect, which pertains to the computational limitations of physical processes. He further granulates the physical aspect into bold and modest versions, which result in the following categorization of the Church-Turing thesis:

Mathematical : any function that is computable by following an effective procedure is Turing-computable.

Bold Physical : any physical process is Turing-computable. This is a concise formulation of *pancomputationalism* – the thesis that everything **is** a computational process (Piccinini, 2017).

Modest Physical : any function that is physically computable is Turing computable.

Further constraints on physical computation which distinguish *useful* computation from all the rest (note immediately the relation to goal-directed and deterministic mode of thinking encoded into the word "useful") are defined:

- **Usability constraint:** if a physical process is a computation, it can be used by a finite observer to obtain the desired values of a function. The sub-constraints of the usability constraint, as summarized by Nayebi (2014) are:
  i. **Readable Inputs and Outputs:** The inputs and outputs of a computation must be readable, in the sense that they can be measured to the desired degree of approximation. For example, having an infinite precision real number for the input and/or output would not be permitted. For an output to be readable in the intended sense, the computing system must have a recognizable halting state.
  ii. **Process-Independent Rule:** In a genuine computation, the problem being solved (equivalently, the function being computed) must be definable independent of the process of solving it (or equivalently, computing it).
  iii. **Repeatability:** For a physical process to be a genuine computation, it must be repeatable by any competent finite observer who intends to obtain its results, in the sense that the same sequence of states can occur either within the same physical system at different times or within two relevantly similar physical systems (e.g. two systems satisfying the same equations).
  iv. **Settability:** An ordinary computing system can compute any value of one or more functions, within its limits (e.g. a universal machine can compute the value of any computable function until it runs out of memory or time).

Moreover, a system must be settable, namely, a user can choose which value of the function is going to be generated in a given case.

v. **Physical Constructibility:** If a system cannot be physically constructed, it may count as performing notional computations, which are irrelevant for physical purposes.

vi. **Reliability:** While the requirement that machines should never break down is unrealistic, it is a requirement that machines be reliable in the sense that they completely operate correctly long enough to yield correct results at least some of the time. A machine that never completes its computation successfully is certainly not worth building.

The above, called a modest physical Church-Turing thesis, very precisely define strictly deterministic processes or what we would call *closed computing*. Consequently, all computation that satisfies these constraints is strictly deterministic while all that does not satisfy them lies somewhere on the right in the continuum of Figure 4.1. The equivalently constrained processes are also called *algorithmic computation*:

> [..] computation that is performed in a closed-box fashion, transforming a finite input, determined at the start of the computation, to a finite output, available at the end of the computation, in a finite amount of time (Eberbach, Goldin, and Wegner, 2004).

4.2.5 Open computing

The requirements for expressing progressive determination are almost orthogonal to closed computing as defined in Section 4.2.4:

i. the readability of inputs and outputs is determined during the process and not *a priori*, which violates the usability constraint of *readable inputs and outputs*;

ii. the function being performed by the process is defined during the process and not before, which violates the *process-independent rule*;

iii. progressive determination is context dependent, non-deterministic and non-directional, and therefore clearly cannot be expressed by computation that satisfies the *repeatability* constraint;

iv. since progressive determination is divergent rather than convergent and the function being computed is determined during the process, it obviously violates the *settability* constraint;

v. the only usability constraint that synthetic cognitive development fully satisfies is the physical constructibility constraint and that is by definition – synthetic cognitive development is the physically implementable computational process simulating the abstract process of progressive determination (see Section 3.2.4);

vi. since synthetic cognitive development and progressive determination are non-deterministic, their "success" or "failure" are poorly defined notions – therefore they cannot satisfy the *reliability* constraint of usability, as it is formulated by Piccinini (2011). Yet, it does not mean that the process can "break down", be "unrealistic" or does not require distinguishing between errors and features in its computational implementation.

Luckily, as discussed in Section 4.2.3, the deterministic and closed computing models are not limits of computation, neither in a theoretical nor practical sense. Turing (1937) had already defined choice machines or "c-machines" which interact with humans or other machines (albeit he did not develop the concept in detail). Turing (1948) defined unorganized machines which are largely random in their construction and structure yet have a defining property of being able to *become organized* into universal machines via learning, evolution or reinforcement learning. These were predecessors of the contemporary paradigm of *interactive computation* which features computing models that allow for changing specification of the machine and adding additional information to initial input during the computation process (Goldin and Wegner, 2006; Milner, 1993a).

Unorganized machines

Turing (1948) observes that the argument of the incompleteness theorem and Church-Turing thesis (Church, 1936b; Gödel, 1931; Turing, 1937) rests on the condition that a machine should not make mistakes, but this requirement is not realistic for a real-world intelligence. Then Turing (1948) describes logic computing machines, practical computing machines, their universal versions, paper machines and partially random machines just to arrive at the concept of the *unorganized machine*. As the name suggests, an unorganized machine is a computing engine that does not have a systematic structure at design and is largely random in its construction.

The machine is made up of a large number of similar units connected via inputs and outputs. An example of such unit could be AND or NAND gate, yet in general they can be made to perform any arbitrarily chosen operation (see Figure 4.2). Connections that relate inputs and outputs can be:

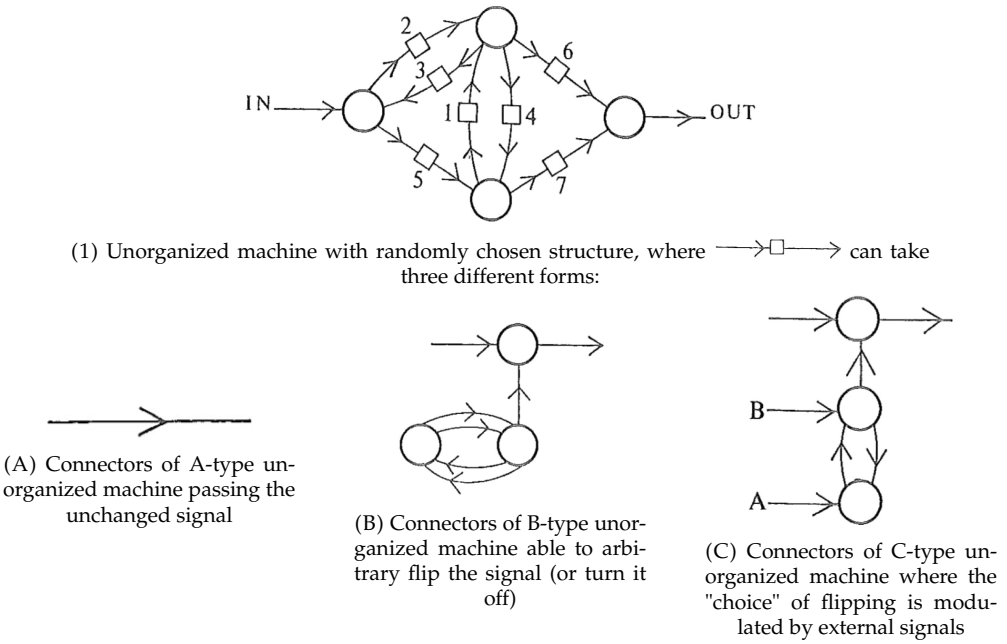
A) fixed from the beginning randomly (see Figure 4.2A),

B) same as A) plus the ability to arbitrarily "choose" to perform NOT operation on the signal passing through (see Figure 4.2B),

C) same as B), except that the "choice" is not random but modulated by external signals (see Figure 4.2C)

The most important property of unorganized machines is that they "have configurations such that once that configuration is reached, and if the interference thereafter is appropriately restricted, the machine behaves as one organized for some definite purpose" (ibid.). They can be called "meta-universal" due to holding a potential to be turned into the configuration of a universal machine as defined by Turing (1937). Therefore, it is not the initial or immediate configuration that is important, but the very process of *organizing the unorganized machinery* during which unorganized becomes organized.

The concept of unorganized machines and the process of their organization were forgotten and later independently rediscovered as neural networks, evolutionary computing and reinforcement learning (Eberbach, Goldin, and Wegner, 2004). Interestingly, at the end of his report Turing (1948) discusses briefly two aspects of educating machinery for organizing the unorganized: *discipline* and *initiative*. Reinforcement learning, or the then so called "pleasure-pain system" is a form of strict discipline which amounts to rewarding for good behaviours and punishing for bad



(1) Unorganized machine with randomly chosen structure, where → □ → can take three different forms:

(A) Connectors of A-type unorganized machine passing the unchanged signal

(B) Connectors of B-type unorganized machine able to arbitrary flip the signal (or turn it off)

(C) Connectors of C-type unorganized machine where the "choice" of flipping is modulated by external signals

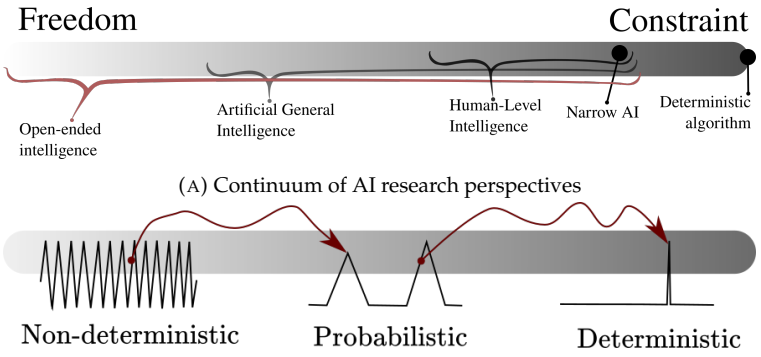
FIGURE 4.2: Types of unorganized machine and its connectors. All pictures adapted from Turing (1948) by rearranging for a concise representation.

behaviours as deemed proper by creators and educators of a machine. "But discipline is certainly not enough to produce intelligence. That which is required in addition we call initiative. This statement will have to serve as a definition. Our task is to discover the nature of this residue as it occurs in man, and to try and copy it in machines" (Turing, 1948, p. 25(125)). Two methods of inoculating the initiative into a machine are given, both of which considered legitimate: (1) designing or pre–training a fully "disciplined" machine and then "grafting in" the initiative by progressively relaxing machine's constraints or (2) starting with a completely unorganized machine without designed constraints and trying to bring into it both "discipline" and "initiative" progressively, yet simultaneously.

The history of the artificial intelligence research programme has clearly demonstrated in many different ways through trial, failure, "summers" and "winters" that no matter how precisely rules of behaviour are encoded into an engine, however complex the learning algorithm is or however well the training set is crafted to cover all examples, something – the component of "initiative", "choice" or context dependency – is missing and that makes learning machines fail miserably where humans or animals excel without effort from a very young age. Remarkably, most of AI architectures and approaches developed through its history unmistakeably follow the first method "of inoculating the initiative" and almost none follow the second, neither conceptually nor implementation-wise.

Conceptually, the two methods of introducing initiative into a machine can be approached through the *freedom and constraint* principle (see Section 2.1.5), by replacing "discipline" with "constraint" or "goal-directedness" and "initiative" with "freedom" or "open-endedness". The predominant way of thinking about intelligence and, subsequently, AI and its implementations is constraint-based: it starts by pre-defining

the goals and behaviours with which intelligence should keep in line. These constraints are relaxed only after observing that they are not enough to produce intelligence and even then are still for the sole purpose of achieving pre-defined goals in the first place. Open-ended decentralized computing takes the opposite path – it starts from the unconstrained ("unorganized") machine and via the process of progressive determination of constraints reaches behaviours that are complex, intelligent and observably useful.



(A) Continuum of AI research perspectives

(B) Continuum of probabilistic behaviours (copied from Figure 4.1)

FIGURE 4.3: Relation between freedom and constraint continuum of Section 2.1.5 (page 14) and non-determinism and determinism continuum of Figure 4.1 (page 91). The continuum between "discipline" and "initiative" can be visualized along the same axis. Constrained behaviour is a special case of unconstrained free behaviour.

I believe that the second method of organizing the unorganized is superior even if one takes into account only a cold technical perspective towards AI architecture design. Recall that freedom and non-deterministic behaviours are more general than constrained and deterministic behaviours. Moreover, we do not know what intelligence is or what constraints define it. Therefore, it makes perfect sense to design a constraint-free architecture that is able to take different constraints as parameters rather than designing specific architecture for every hypothetical constraint that is deemed to be important to intelligent behaviour. Metaphorically speaking, the rationale of designing a computational framework and architecture without positing any constraints is not much different from the rationale of designing a universal (Turing) machine rather than every specific computing engine. Moreover, it seems the unavoidable design choice for an architecture that can change and self-modify its computation processes and configuration as well as using intermediate results for determining further processes.

The image of the unorganized machine of Turing (ibid.) clearly follows the network paradigm (see Section 2.2.4) by allowing elementary computation units to connect via configurable connectors and organizing network structure in this way. A modern computational paradigm following the same principles is *interactive computation*.

Interactive computation

Interactive computation (Dina Goldin, 2006; Milner, 1993a; Wegner, 1998) extends Turing (1937) machines by adding dynamic input and output (read and write) actions that interact directly with an external environment which may be composed of

other interaction machines. It can handle inputs and actions occurring during computation additionally to the input that is defined before computation starts, in this way providing history-dependent results and services.

The model of interactive computation is arguably super-Turing and the only physically realizable hyper-computation model (Nayebi, 2014). Formal models of interactive computation include the communicating sequential processes (CSP) of Hoare (1978), the process calculus (π-calculus) of Milner (1993b), the actor model of Hewitt, Bishop, and Steiger (1973), the cost-calculus (\$-calculus) of Eberbach (2000) and more. These models have been developed at least partially in order to account for the need to reason about parallel and concurrent computing, for which the refinement of sequential computing paradigm was not sufficient (Milner, 1993a).

By the 1970s it was already realized that classical Turing machines do not account for all problem solving, but devising a complete model of computation that includes interaction was not attempted until the 1990s (Goldin and Wegner, 2006). While the interactive view of computation is widely accepted by many programmers and realized in software development frameworks, such as actor framework, reactive programming and event-driven service-oriented architectures[4], it is disputed by adherents of the classical Turing machine model who regard interaction, super-Turing computation or hyper-computation at best an unproven and unnecessary paradigm shift (Davis, 2004).

For the purposes of this work it is enough to admit the superior expressibility and practicality of some super-Turing computation models and their implementations, such as interactive computation. But it is still interesting to note the conceptual aspect of the debate on the possibility of computational models extending that of Turing (1937). Referring to the Gödel (1931) incompleteness theorem, Goldin and Wegner (2002) show that interactive and empirical computation are *incomplete* because they have too many variables following chaotic dynamics to be expressible as theorems of a sound and complete logic. They further suggest that interactive computation can be formalized in *paraconsistent*, i.e. inconsistency-tolerant, logic. Turing's 1937 machine was conceived to be intentionally formalizable and consistent in order to solve the *halting problem* in mathematics. It seems therefore that the debate of hyper-computation again revolves around deterministic versus nondeterministic and a "desire for predictability of the world" and involves a kind of "Gödelean choice" between incomplete and inconsistent.

4.3 Conundrum of decentralization

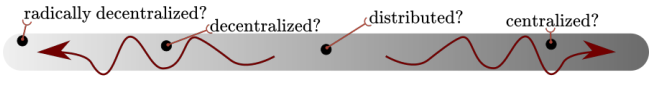
Decentralization in its most basic definition means the process of transfer of authority from central to local government[5] and is historically mostly related to social governance. The concept has been applied in management (e.g. decentralized organization structures), political science and ideology (e.g. liberal social decentralization, anarchy), economics (e.g. free market), computing and technology (e.g. internet,

[4] (1) CAF_C++ Actor Framework (2) Implementation of actor model on the Java VM (3) Actor framework for Rust programming language (4) Cross-platform Actor framework (5) Asynchronous programming with observable streams (6) Event-driven SOA

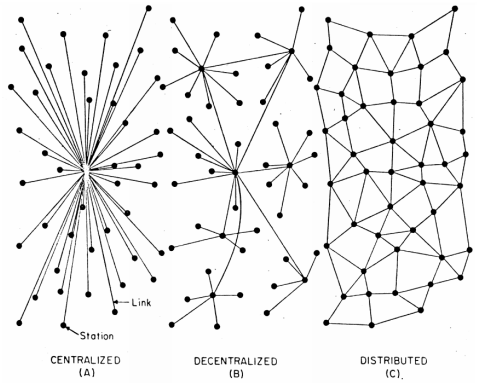
[5] https://en.oxforddictionaries.com/definition/decentralization

blockchain) and much more. Due to the many domains of usage, the term has obtained so many different meanings and connotations that it has become uncomfortably vague and poorly defined, especially in the socio-technological domain (Buterin, 2017). Yet, despite their broad usage, concepts of centralization, distributedness and decentralization are general systemic properties. We will now attempt to provide system-theoretical treatment of the decentralization concept with the purpose of clarifying its role in the computational model.

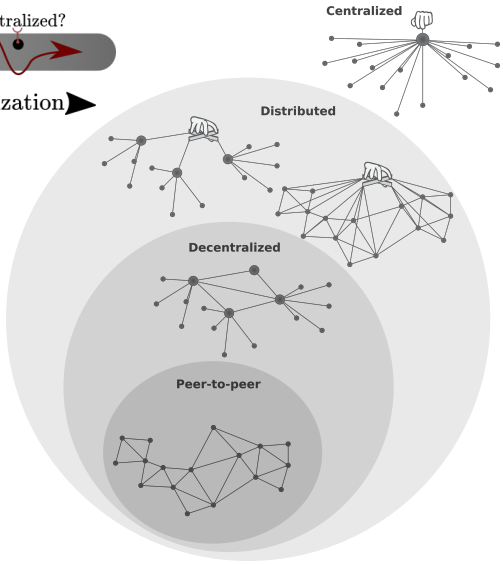
First, centralization and decentralization are aspects of the topological configuration or structure of a system and are best approached from the perspective of network science (see Section 2.2.4), which allows the structure and dynamics of systems to be abstracted from their immediate domain (see Figure 4.4). This way it is possible to describe a system in terms of nodes (denoting objects of the system) and links (relations between objects). Note that links can represent any dynamic or static relations – control, information flows, friendship, etc.



(A) Continuum of de-centralization



(B) Centralized, decentralized and distributed networks. Adapted from Baran (1962, 1964)



(C) Relations among centralized, decentralized and distributed. Adapted from MaidSafe (2015)

FIGURE 4.4: Defining the continuum between centralization and decentralization in terms of network structures.

The language of network science allows the notions of centralization, decentralization and distributed-ness to be defined more clearly. Note, however, that historically the usage of some of these terms has been inconsistent (see e.g. the different signification of "decentralized" and "distributed" in Figure 4.4) and therefore our definitions are driven by conceptual clarity of intended usage rather than historical usage (Figure 4.4c):

- a system is **centralized** when one "central" node or actor is directly linked to all other nodes. Formally, given a network of N nodes, there exists only one node in a network with degree equal to $N-1$ where multiple links to the same node are counted as one;
- a system is **distributed** when there is one "central" node from which all other nodes are reachable (i.e. a path through other nodes exists), yet they are not necessarily directly connected;

- in a **decentralized** system there is no single central node, but a few, which provide shorter direct or indirect paths for different sub-networks of the whole network;
- a **peer-to-peer** system is where all nodes have the same degree – in either a fully or sparsely connected network[6].

A few notes are in place here. First and obviously, mapping these concepts to well defined network structures allows one to reason about properties of a system as well as providing sound measures for estimating them. Second, most of these structures and concepts do not exist in their pure form since real-world graphs always exhibit some combination of them (see Figure 4.4a). Third, a network is a model of a real system, and therefore links between nodes represent arbitrarily chosen aspects of relations between them. Which particular types of relation are chosen is the subject of the *selection for relevance* aspect of model building (see page 29). Recall, however, that there exist network and graph formalisms which allow different relation types to be represented in the same graph (see page 40). A so called multi-labelled network representation of a system may be centralized with respect to some types of relation and peer-to-peer with respect to others. This is often the source of confusion when using a single term to describe real world systems without naming the aspect of analysis – see Box 4.1 for a *blockchain* related example.

Box 4.1: Distributed consensus and decentralized aspects of blockchain technology

Blockchain is the data structure of a growing list of records composed into blocks which are cryptographically linked ("chained") to each other in a manner that defines total order between them, developed by Haber and Stornetta (1991). The first conceptualization and implementation of blockchain by Nakamoto (2008) featured an important addition of a model which does not require an exogenous trusted party, but is used to achieve an *eventually consistent* state of the data structure without a centralized mechanism. This conceptualization provided a practical solution to the fundamental *Byzantine Generals' Problem* in computer science and, particularly, distributed computing systems (Lamport, Shostak, and Pease, 1982). The problem is formulated in the context of generals at different camps outside the enemy castle deciding whether to lead their armies to an attack. The attack can be successful only if all armies act in a coordinated manner. Generals have no other way to communicate except by sending messengers through enemy territory. Both messengers and commanders of the armies can be traitors and there is no way to know it – there is no "omniscient" entity or a party whose trust is externally established. The problem of successful coordination of actions in this setting solves the *distributed consensus* problem.

The Bitcoin blockchain of Nakamoto (2008) as well as some other implementations, including Ethereum[a] cannot be unambiguously described as decentralized or centralized from a systemic perspective, since they combine both, depending on which type of relationship is considered:

- they are **centralized** if we model nodes as units of information and links

[6] Equivalent definitions can be used for discrete or weighted links between nodes.

as exchange of information – since there is a single global data structure which is accessed by everybody in the network;

- they are **decentralized** if nodes represent individuals and organizations controlling computing resources of the network;
- finally, in terms of distribution of physical machines and the probability of their fault, blockchain systems can be **peer-to-peer**.

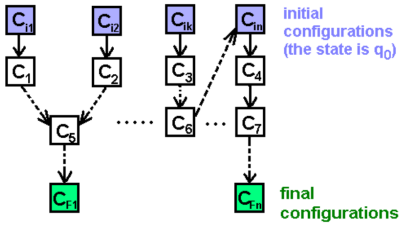
The above listed aspects were named *logical*, *political* and *architectural* decentralization by Buterin (2017) and are characteristic of blockchain technology. Yet the principle is generalizable to any system with an arbitrary number of aspects of analysis. The overall dynamics and properties of a system would depend on actual configuration along these dimensions as well as their interaction. Design, maintenance and governance of these configurations are the subject of what we call *decentralized IT governance*.

[a] Ethereum White paper: A Next-Generation Smart Contract and Decentralized Application Platform.

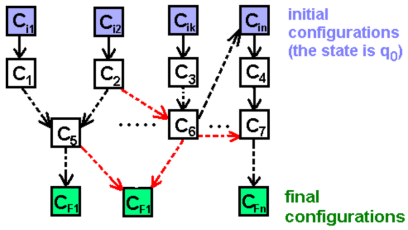
Apart from being based on blockchain technology, Ethereum is the first system constituting a so called "smart contract and decentralized application platform", essentially running a single virtual machine on a decentralized network of computers able to perform Turing complete computation by maintaining the eventually consistent global state and history of its operations. By greatly simplifying and abstracting from details, the platform is conceptually modelled after the deterministic model of Turing (1937). For connecting the platform to the real world the notion of oracles are used – which, just as in the modified Turing (1938) model, allow for non-deterministic inputs into otherwise deterministic computation.

This hints at a relation between non-determinism and decentralization, which does not finish here. First note that any finite computation or a program can be expressed as a directed graph with atomic processes as nodes and transitions between them as links (Allamanis, Brockschmidt, and Khademi, 2018; Emde Boas, 2012; Goertzel, Pennachin, and Geisweiller, 2014; Rodriguez, 2010; Turchin, 1986; Turing, 1948). A *computation graph* is a structure where nodes represent atomic computation units and links – both control and data flows between them. A deterministic computation can then be represented as a simple graph where each process node has only one outgoing link to the next process node (Figure 4.5a). Likewise, a non-deterministic computation is a simple graph where a process node can have more than one outgoing link where each link is "chosen" at runtime by the control flow in probabilistic manner (Figure 4.5b). Further, a non-halting deterministic computation can be expressed as a graph with a cycle which is unfolded to a simple graph when the computation is stopped and the number of times the control flow of computation has taken the cyclic path is known (Figure 4.5c).

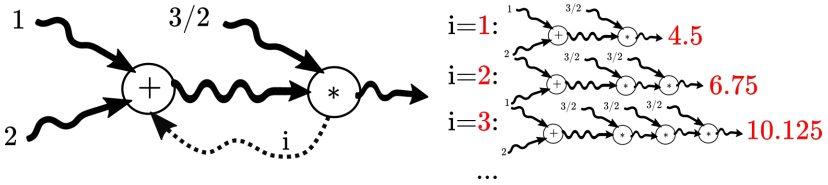
Figure 4.5 illustrates that a finite non-deterministic computation can be reduced to deterministic by pruning away links representing control flow in order for each process node to have only one outgoing link. Alternatively, a deterministic process can be represented in a non-deterministic computation graph if the process is made to know *a priori* which outgoing links it will take at each node. In other words, *a deterministic computing graph is a reduction of a non-deterministic computing graph* and clearly the former is a special case of the latter. Therefore, it is possible to achieve a



(A) Deterministic Turing machine graph. Adapted from Laud, 2011.



(B) Non-deterministic Turing machine graph. Adapted from Laud, 2011.



(C) The graph of an arbitrary trivial function $(x + y) \times \prod_{i=1}^{\infty} z$. The geometric progression is represented by the dashed line in the graph on the left, but when i is known (i.e. the computation is finite), it can be "unfolded" to a graph on the right. In a general case the computation may never end, but intermediate results can be *read out* after an arbitrary number of steps or time units.

FIGURE 4.5: Representing Turing machines as graphs.

deterministic computation from a non-deterministic by imposing a global perspective on every computation process represented in a graph. This is precisely the relation between decentralization and non-determinism: *a deterministic computation is necessarily centralized by the existence of a global observer having control over all and every elementary computation process in terms of their relations to the overall computation graph.* Conceptually, non-determinism reflects uncertainty and disorder from the perspective of a global observer, while determinism – certainty and order. Order is progressively achieved from disorder by breaking a symmetry of uniform transition probabilities among elementary processes in a systematic way. In the language of computer science:

> Distributed [decentralized] computing arises when one has to solve a problem in terms of distributed entities (usually called processors, nodes, processes, actors, agents, sensors, peers, etc.) such that each entity has only a partial knowledge of the many parameters involved in the problem that has to be solved. While parallel computing and real-time computing can be characterized, respectively, by the terms *efficiency* and *on-time computing*, distributed [decentralized] computing can be characterized by the term *uncertainty* (Raynal, 2013)[7].

Table 4.1 illustrates relationships between computing systems, computational models, programming paradigms and execution mechanisms from the (de)centralization perspective. Centralized systems are based on the worldview with an omniscient observer, require deterministic or probabilistic computing, sequential or parallel programming paradigms, synchronous execution and can be explained by the Turing (1937) computation model. Decentralized systems, on the other hand, encompass a worldvew without the omniscient observer, indeterministic computation, distributed programming paradigm, asynchronous execution and can be explained by the model of interactive computation.

[7]Text in brackets added by me to account for ambiguity of terms "distributed" and "decentralized", as explained in Figure 4.4

Systemic perspective	Centralized	Distributed	Decentralized
Computation	Deterministic	Probabilistic	Non-deterministic
Control flow	Sequential	Parallel	Concurrent
Execution	Synchronous	Bulk synchronous parallel	Asynchronous
Model	Deterministic Turing	Non-deterministic Turing	Interactive computation
Communication	Shared medium	Broadcast / multicast	Peer-to-peer message passing
Memory	Local	"Glocal"	Global
View access	Omniscient observer	Powerful observers	Partial observers

TABLE 4.1: Centralization, distributed-ness and decentralization mapped to systemic perspectives and aspects of computing. Note the fuzzy boundaries between the concepts and their properties.

4.4 Consensus and synchronization

Synthetic cognitive development (see Section 3.3.3) is fuelled by the emerging functional coordination of interacting agents in the forms of synchronization, coherence and compatibility. Functional coordination, emergent or designed, allows for interacting actors to structure their internal processes and information exchanges between them in time and space in order to achieve higher levels of synergy (Heylighen, 2013). In the context of computer science and decentralized systems, functional coordination is reflected by the problem of distributed consensus (see Box 4.1) and emerging network structures – i.e. "organizing the unorganized".

Recall that in the context of open-ended intelligence progressive determination of constraints, assemblages and boundary formation are described in terms of a population of initially independent interacting heterogeneous agents (see Section 2.1.6 and Figures 3.12, 3.4 or 3.5). This context is analogous to the one of decentralized computation where computation is achieved via synchronization of independent elementary processes. Therefore the computational aspect of progressive determination of constraints and synthetic cognitive development is tightly related to the research of distributed consensus in computer science. Notwithstanding, there is one important differentiating aspect of the conceptual and computational perspectives: in computer science the "fuzzy" consensus is usually not considered or even allowed, while conceptually, perfect coordination and complete determinism is an exceptional case of human engineered systems.

This relation between conceptual and computational perspectives gives rise to the following considerations, which together form the core of the computational model of open-ended intelligence:

i. A preindividual, consisting of a non-coordinated population of agents (Figure 3.4) is comparable to a Turing (1948) "unorganized machine" (Section 4.2.5) which is in turn analogous to a decentralized collection of processes asynchronously and randomly communicating among each other. Such a framework, while it can be perfectly well implemented in a computational medium, initially produces no more than a random noise. Moreover, there is no clear "input" or "output" of the computation as these can be any agent of the population.

ii. Alternatively, a fully formed individual is, somewhat simplifying, comparable to a centralized synchronous meta-process assembled from separate processes into a deterministic computation graph (Figure 4.5a);

iii. A fluid individual is, again with a chunk of salt, comparable to a probabilistic computation graph or a non-deterministic Turing machine (Figure 4.5b), where patterns of communication between elementary units of computation exist, but are probabilistic and fluid rather than strictly defined.

Note that the above analogies do not claim strict isomorphism but are rather guiding principles for proposing an implementable computational model and later software architecture informed by the concept of open-ended intelligence and synthetic cognitive development. In terms of these considerations, the synthetic cognitive development is manifested by the emergence of repeating communication patterns among elementary computational units – i.e. formation of deterministic or probabilistic computation graphs from non-coordinated communications of initial population of elementary communicating units. Further, a deterministic computation graph of elementary units can itself be considered a unit of computation at a higher scale. As such, it can participate in the emergence of a "meta-computation" graph, where meta-patterns of communication between higher scale units come about in the same manner. Such multi-scale computation graph is analogous to the model of scales of individuation (see Section 3.3.2).

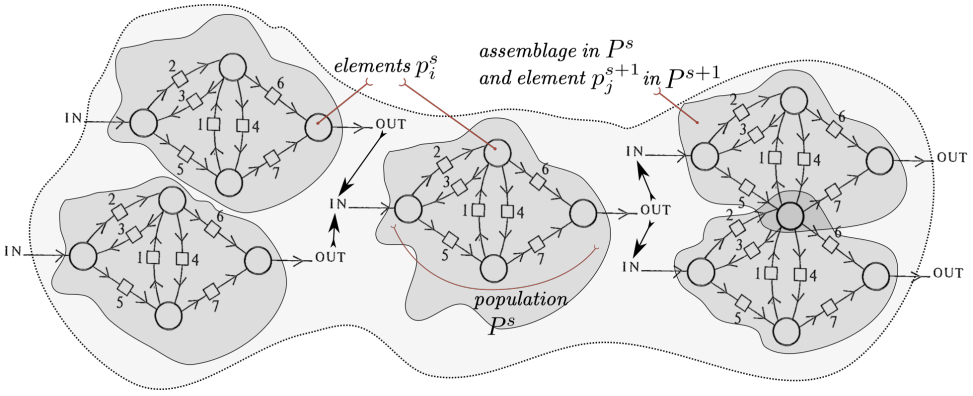


FIGURE 4.6: Scalable structure of a computation process as a structured assemblage of assemblages of lower level computation processes.

The emergence of patterns and meta-patterns of communications between elementary computational processes is a result of functional coordination that in a computational context is best expressed by the notions of synchronization and consensus. Adaptively changing levels of synchronization and consensus of a decentralized computational system manifests the scheme of synthetic cognitive development (see Figure 3.14) of a system and an open-ended computational process. In order to move forward with the formulation of the main tenets of the model of open-ended decentralized computing in Section 4.4.1, we first briefly define the notions of synchronous and asynchronous computing and consensus. Then different computing systems in terms of their level and sources of synchronization and consensus are discussed.

The difference between synchronous and asynchronous computation and execution becomes important when dealing with many independent processes which try to coordinate their actions and behaviour by interacting with each other. Such

coordination is a necessary component of decentralized and distributed systems, which are not limited to computational systems only. For example, the human social system is also a decentralized system where independent heterogeneous actors communicate, form assemblages, coalitions, nations, societies, families and more or less hierarchical organizations (Veitas and Weinbaum, 2017). Another example is the so called internet of things (IoT) – a network of cyber-physical systems connected via the internet – which is estimated to outgrow in number humans, mobile phones and other devices by orders of magnitude as of 2025[8]. The sheer scale and complexity of these systems will require the adoption of the systemic perspective of decentralized governance – social, IT or other kind. Most of the communications in these systems will happen, not between humans and humans, or humans and devices, but between devices and devices. The socio-technological system will no longer be able to operate on the basis of humans deciding what needs to be done and then instructing devices to carry out certain actions. Devices, as loosely coupled computational systems, will have to decide on their individual behaviours by coordinating with other devices, occasionally asking for help from humans or providing already processed and reduced information to them. Even if individual devices would manifest completely deterministic and synchronous processes, the very fact of their interconnectedness into one network and constant interaction would make the global system non-deterministic, manifesting the model of interactive computation (see page 97).

A population of independent interacting agents is best expressed within the domain of computer science in terms of a message-passing framework (Hewitt, 1976), where synchrony and asynchrony are clearly defined (Raynal, 2013):

- **Synchrony.** A communication among processes is considered *synchronous* when every process observes the same order of messages within the system. In the same manner, the execution is considered *synchronous* when every individual process in the system observes the same total order of all the processes which happen within it.
- **Asynchrony** is the reverse of synchrony. A communication among processes is considered *asynchronous*, when every communicating process can have a different observation of the order of messages being exchanged (e.g. they can receive the same message at different times). Likewise, the execution is considered *asynchronous* when there is no single established way for processes to determine the total order of the execution of all processes within the system. Put in yet another way, there is no notion of exogenous global time or global perspective in asynchronous systems – these notions have to be *agreed* by participants within the system.

Note immediately that synchrony is related to centralization and determinism, while asynchrony to decentralization and non-determinism. Not surprisingly, any synchronous communication system can be built on top of an asynchronous communication system (Agha, 1986, p. 19) by constraining the latter. Synchronization is a process of introducing constraints to the asynchronous system so that it gradually becomes synchronous. In this sense, achievement of distributed consensus and the progressive determination of constraints are both mechanisms of synchronization. This leads us to the key observation of this chapter, that the mechanism of progressive determination, central to the notion of open-ended intelligence and synthetic

[8]https://www.businessinsider.com/iot-forecast-book-2018-7?IR=T

cognitive development, can be expressed computationally in terms of synchronization (and de-synchronization) of communications in a message-passing system.

The difference between synchrony and asynchrony is applicable to and helpful for reasoning about general decentralized systems. Natural asynchronous systems abound – nature itself can be considered an asynchronous decentralized system. For example, as established by Einstein's theory of relativity, two astronomical events can be observed in different order by two independent observers if each of these observers is closer to a different event and if there is no notion of global time. Clearly, without any level of synchrony (global or local) within a system it is quite difficult to reason about it and impossible to build a model. Nevertheless, complex adaptive systems cannot be purely synchronous or purely asynchronous. In the former case, they become rigid rather than adaptive and complicated rather than complex. In the latter – purely asynchronous – case, there is no system to reason about in the first place. The most interesting systems are those that exhibit certain aspects between synchrony and asynchrony and, most importantly, are able to adaptively and non-trivially change this balance. Such systems manifest open-ended decentralized computation.

4.4.1 Open-ended decentralized computing

Finding a balance between synchronous and asynchronous aspects of an execution and a way to specify constraints for achieving synchronization in a manner that allows for a computational system to be flexible and adaptive enough, while at the same time able to produce observably interesting and useful results, is the main challenge of open-ended decentralized computing. It is clear that in order to build a pragmatically viable computation system, some external constraints will have to be designed and initiated initially. It is also clear, especially with respect to generally intelligent systems, that behaviours not directly related to these initial constraints, but born from the flexibility of progressive determination during the system's operation, may be more important to the overall performance than the initial set-up.

Conceptually, open-ended decentralized computing supports a paradigm shift in computer science resembling that of developmental systems theory (DST) in biological development, heredity and evolution (see page 37). Recall that DST views development and evolution – i.e. individuation – of organisms as a "process of *construction* and *reconstruction* in which heterogeneous resources are contingently, but more or less reliably reassembled for each life cycle". Likewise, open-ended decentralized computing effectuates computation as the processes of *integration* and *disintegration* in which heterogeneous independent processes probabilistically but reliably are assembled and reassembled to pragmatically useful "goal-directed" computing graphs. This process, fuelled by internal and external ecological flux, moves fluidly within the meta-stable landscape of local or global synchrony and consensuses as illustrated in Figure 4.7.

The meta-stable landscape of open-ended computing is unbounded and incomputable but we can informally delineate it with descriptions of a few characteristic cases, as discussed further.

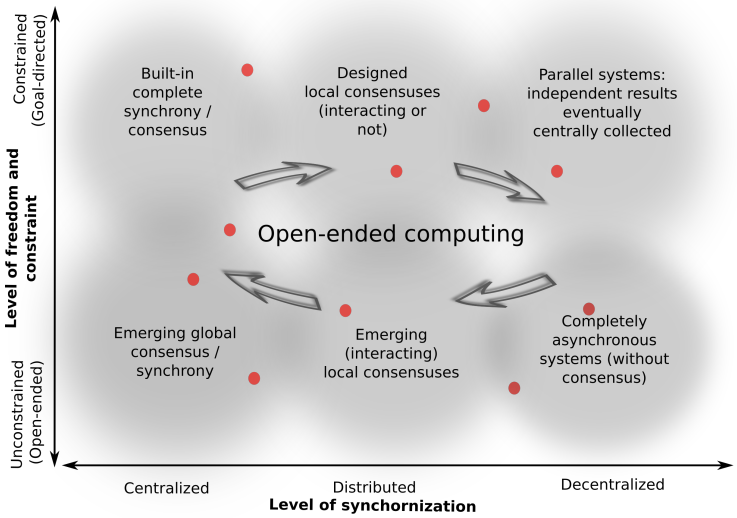


FIGURE 4.7: Open-ended decentralized computing effectuates the self-organized movement of the system's "configuration" within the meta-stable landscape described by the dimensions of *freedom and constraint* (vertical axis) and *synchronization* (horizontal axis). Note that the boundaries between these configurations are fuzzy.

Built-in complete synchrony and consensus

Enforcing synchrony onto the system without allowing any asynchronous behaviour is essentially equivalent to designing a deterministic architecture and program. Note that such a system can be distributed (i.e. utilizing parallel processing for trading memory for speed) yet still centralized and synchronous, having a deterministic behaviour, single global clock or shared data structure, which establishes total order on all events happening in such a system.

Completely asynchronous systems

As noted before, completely asynchronous systems cannot be reasoned about, and therefore there are no such systems in their pure form, at least outside research laboratories. Nevertheless, there could be systems that are close to completely asynchronous in their design. For example, the internet could be considered such a system by design, where some local synchrony and consensus eventually emerges. For example, when two computers start negotiating a TCP/IP connection they can be considered asynchronous with respect to each other, but get synchronized when connection is established. Complete asynchrony without the mechanism for emergent consensus is not practical by any measure, but an open-ended decentralized computing system may transitionally find itself in a state close to such.

Emerging global synchrony

A viable approach of designing initially asynchronous systems is to include a mechanism requiring a system's components to reach synchronous behaviour without explicitly enforcing consensus from the start. Systems based on blockchain technology (Box 4.1) are examples of this category. Here, synchrony is achieved by eventually

achieving a global data structure through which all participants of the system access the same total order of events.

Blockchain technology based systems, introduced by Nakamoto (2008), are particularly interesting for their hybrid nature – they achieve global consensus not only via technical means, but also with the help of economic and game-theoretic mechanisms. The Bitcoin example reveals four constituencies involved in reaching consensus (Antonopoulos, 2015):

1. **System creators** implement process consensus via software design, update proposals, rule changes, etc.;
2. **Trust providers** implement runtime consensus by effectuating the system's consensus protocol – by voting for the longest chain with the computing power at their disposal in the case of "proof-of-work", with assets in the case of "proof-of-stake" or with reputation score in the case of "proof-of-reputation" consensus protocols (Kolonin et al., 2018);
3. **Liquidity providers** – cryptocurrency exchanges – make the system open by allowing participants to enter and exit it;
4. **Marked participants** – wallet holders, businesses and merchants – provide incentives and directions of development to other constituencies by using the system.

The above list emphasizes that the overall consensus in a blockchain-type system is achieved not solely via trust providers, as is often assumed, but via a complex ecosystem of consensus layers. One can imagine systems where different constituencies enjoy different importance weights in achieving consensus[9]. In a manner of speaking, a system with built-in synchrony is one where system creators have 100% voting power for the initial consensus setting and where consensus is not allowed to change. The configuration of a system involving separate but interacting consensus layers is a general aspect of a decentralized IT governance, not limited to blockchain technologies.

Emerging local synchrony and consensuses

Global synchrony and eventual consensus is not needed or even desired in all decentralized systems. For example, in the case of the internet, one does not need to ensure that all data packets and messages in the network are well ordered: only packets pertaining to higher order events (e.g. a client-server session or a conversation) have to be strictly ordered in order to ensure the operation of the network. In other words, only some elements of a system have to be synchronized and only for a limited period of time. All other participants can stay asynchronous with respect to each other until the need to coordinate arises. Normally, such a network would at every moment support a number of local consensuses and synchronized elements which change fluidly over time as the situation requires[10]. Moreover, local consensuses may interact among each other for guaranteeing certain global properties of the system. In order for such a model to work, the system has to provide a mechanism for local consensuses to emerge when needed and dissolve afterwards.

[9] For example, in Ethereum ecosystem, system creators have much more importance than in Bitcoin community.

[10] e.g. *Holochain* https://holochain.org/

Note that the model of emerging local consensuses is an exemplar of open-ended decentralized computation architecture which is fluid enough to be able to morph into other models. For example, if a local consensus in such a system grows to encompass and synchronize all elements of the system, it may become a global consensus. Likewise if all local consensuses disintegrate it would become an asynchronous system. Most often, however, such a system would balance between these extremes.

Designed local consensuses

In certain cases it may make pragmatic sense to "help" the system to achieve global coordination by designing and "locking" local consensuses as part of system's architecture. This may put more constraints on systems behaviour, and therefore make it less fluid yet more predictable[11].

Parallel systems

Technically, according to definitions proposed in Section 4.3 and Box 4.1, parallel systems are architecturally distributed and logically centralized. The power of such systems and model of computation comes from the ability to split the overall task of computation into smaller non-communicating elements of comparable complexity, perform them on different physical or logical architectural components (servers, processors, threads) and then collect them into the single final result. Examples of computational models of this sort are *MapReduce* (Dean and Ghemawat, 2004), *split-apply-combine* (Wickham and others, 2011) and *gather-apply-scatter* (Gonzalez et al., 2012). While these systems feature different computational elements, they are not independent and are controlled by single logic – therefore closest to the in-built global consensus model.

4.5 Stigmergic computing

As was introduced in Section 2.2.4, we are going to use the principle of *stigmergic computation* in order to implement processes that drive the movement of an open-ended decentralized computing system in its meta-stable configuration space (see Figure 3.9 for the notion of degrees of meta-stability). Stigmergic computing is a computational model of generalized stigmergic cooperation, which is an indirect coordination between independent actors via a shared medium, where some actors leave a trace that can be picked and acted upon by other actors. In this way, stigmergic computing enables the mechanism of progressive determination by providing a way to relate the actions of individual elements to the overall structure and function of the system. It realizes both effects of *emergence* (a novel behaviour arising from lower scale of a system) and *immergence* (an individual action informed by the higher scale of a system) in the single framework (Marsh and Onof, 2008). Such a system is thus able to achieve synchornization and consensus modes as described in the previous section and enable the mechanism of progressive determination (see Section 3.2.4).

[11] e.g. *SAFE Network* https://safenetwork.org/

Recall, that progressive determination formally, yet very generally, can be represented as a chain of transitions between structure and operation, where each operation changes the structure and each structure changes the operation: $...S_1 \rightarrow O_1 \rightarrow S_2 \rightarrow O_2 \rightarrow S_3 \rightarrow ... \rightarrow O_n \rightarrow S_{n+1}...$ For the implementation in a computational medium we have to represent it in a much more detailed manner. First, in a decentralized setting, the operation $O_1, O_2, ..., O_n$ is an abstract representation of the totality of many heterogeneous operations of independent processes in a population. Second, every sub-process $O_n \rightarrow S_{n+1} \rightarrow O_{n+1}$ requires an intermediary structure to be accessible for both reading and writing, which, considering asynchrony, can happen at the same time. Consequently, third, the structure $S_1, S_2, ..., S_n$ is decentralized in the sense that every process that realizes an operation should be able to access only part of it without influencing ("locking") other parts. Fourth, the system should be able to operate in a situation in which no single process can reference the *whole* structure either for reading or for writing. Actually, in a normal operation, a single process would be able to access only a small specifically localized portion of the data structure. Graphically, the refined representation of progressive determination as stigmergic computation is illustrated by Figure 4.8.

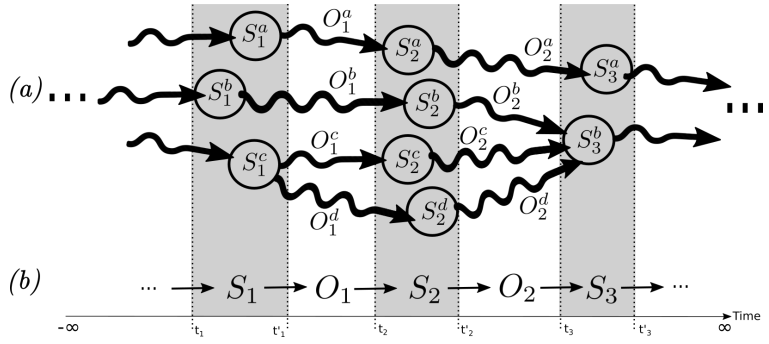


FIGURE 4.8: Stigmergic computing. (b) is the formal representation of the progressive determination process in terms of operation giving rise to structure and structure giving rise to operation; (a) is the visual representation of underlying asynchronous and concurrent operations which change parts (overlapping or not) of the structure; (b) can be inferred by an observer of a system by setting time granularity of observations, allowing to reduce $O_n^i, O_n^{i+1}...$ to O_n and $S_n^i, S_n^{i+1}...$ to S_n. Note, that in a reasonably complex system, such reduction necessarily involves loss of information about some (non-essential) states and some operations as described in Section 2.2.3 on page 28 about model building.

In a decentralized computing system, where processes do not have access to the global state of data structure, operations will change the structure by affecting it locally. Likewise, the structure will affect operations via its local sub-structures. As illustrated in Figure 4.8, the general model of progressive determination $...S_1 \rightarrow O_1 \rightarrow S_2 \rightarrow O_2 \rightarrow S_3 \rightarrow ... \rightarrow O_n \rightarrow S_{n+1}...$ is an abstraction over the multiplicity of local processes and operations $...S_{1a}, S_{1b}.. \rightarrow O_1^a, O_1^b.. \rightarrow ...S_{2a}, S_{2b}.. \rightarrow O_2^a, O_2^b... \rightarrow S_{3a}, S_{3b}... \rightarrow ... \rightarrow O_n^a, O_n^b.. \rightarrow S_{n+1}^a, S_{n+1}^b, ...$ This abstraction can be achieved by an observer following the principles of model building and attempting to impose total order on all events happening in the system – i.e. every local operation reading from and writing to a local structure. In order to do that, such an observer has to arbitrarily establish a time scale and granularity of observations, making inferences $S_n \dashv S_n^i, S_n^{i+1}...$ and $O_n \dashv O_n^i, O_n^{i+1}...$ valid. Note that this operation of observation

is a case of lossy information compression (see page 32) subject to a *degree of decomposability*, *selection for relevance* and *structural efficiency* aspects of model building (see page 28).

Note that in computational terms operation is realized by a *process*, while structure by a *data structure*; therefore in our context these terms mean the same and are used interchangeably. The computational model has to accommodate and describe requirements for both these components separately as well as for implementation of their interaction. The remainder of this section describes these requirements in general computational terms in order to provide the basis for the following Chapter 5. Along the way, many concepts introduced in the preceding chapters are defined more specifically using computer science terminology.

4.5.1 Process

Let us first specify the notions of process and agent and their usage in our context. A **process** (or operation) is a computational entity that reads in certain data as an input, performs a transformation on that data using certain function and writes it out as an output (Figure 4.9):

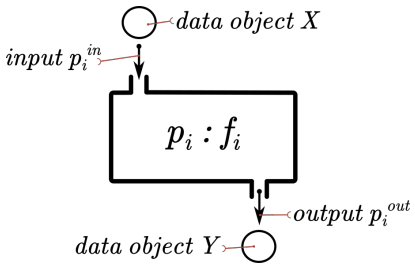


FIGURE 4.9: Illustration of an elementary process p_i as an entity performing function f_i on input (data object o_x) and producing output (data object o_y).

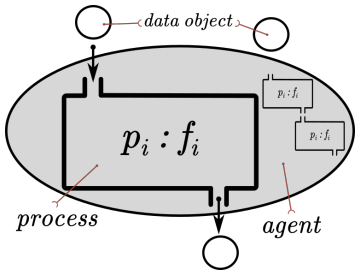


FIGURE 4.10: An agent is composed, or "owns", one or more simple or complex processes by providing computational capacities and energy to perform work of transforming input data objects to output data objects.

An **agent** is an entity which provides resources needed for the process to do its "work", as defined by its function and specific output and input (Figure 4.10). In computational sense such resources are equivalent to computational work given computational capacities as adequate for the complexity of the function. Note that an agent can contain a complex process (i.e. a combination of elementary processes).

4.5.2 Message passing for process interaction

The most suitable computational expression of interacting heterogenous agents is the message-passing system or framework where agents exchange data objects. A *message-passing system* is a system where any pair of processes (p_i, p_j) can directly communicate by p_i depositing (sending) values (messages) in an object (the channel from p_i to p_j), and p_j withdrawing (receiving) values from this object. If we denote such an elementary read/write operation on an object as op, then all operations that

happen during the life of the system produce a set OP. Further, we can define an order operator $\xrightarrow{\text{op}}$ which, when applied to all operations in a set, produces a partial order $\hat{OP} = (OP, \xrightarrow{\text{op}})$. The operator is defined as follows: if we randomly draw two arbitrary operations from OP and denote them op_1 and op_2, then $op_1 \xrightarrow{\text{op}} op_2$ if op_1 terminates before op_2 starts. Two operations which are not ordered by $\xrightarrow{\text{op}}$ are *concurrent*. Note that the partial order $\hat{OP}$ on the set of operations in the system defines a particular computation. If $\hat{OP}$ is the total order, then the computation is *sequential* and synchronized. This is due to Raynal (2013).

By applying the above formalization to the discussion of meta-stable configurations of open-ended decentralized computation (Section 4.4.1) we infer that synchronous, asynchronous or partly synchronous configuration of a system can be expressed as a relative number of totally ordered operations with respect to all operations in the system. Yet this formulation cannot grasp the possibility of partial synchronicity and local consensus(es) without defining the topological relations of operations, processes and objects.

Approached a bit more concretely, every operation in OP is a triple (p_i, rw, X_{ij}), where p_i is the process issuing the operation, X is an object on which the operation is performed, and *rw* is either *read* or *write*. Further, an object X_{ij} is a communication channel between processes p_i and p_j, and therefore is a tuple (p_i, p_j). This allows us to express an operation simply as a relation between processes in the form:

$$op_{ij} = (p_i, rw, p_j), rw \in [\textit{read}, \textit{write}] \tag{4.1}$$

The complete history of interaction $H_{a,b}$ between two processes p_a, p_b in a system composed of N number of processes is then simply a subset of OP:

$$H_{a,b} \subset OP; a, b \in [1, N] \tag{4.2}$$

Likewise, we can define the history of interactions of any arbitrary cluster C of processes with the minimum size 2 and maximum size N as:

$$H_c \subset OP : c \subset N \tag{4.3}$$

Based on the above we formulate working definitions of topological relations and local synchronicity in a message-passing system:

- **Local synchronicity** is the total order of operations of a cluster of processes containing strictly less than all processes of the system (as otherwise it becomes the global synchronicity).
- **Topological relation** between two processes p_i, p_j is defined by existence and frequency of operations op_{ij}, op_{ji} issued by these processes[12].

[12]Consider the brain as an example. Topological relations between neurons in the brain are expressed as synapses, which, on the one hand, are vehicles of communication signals travelling from neurons. On the other hand, synapses grow and get shaped due to this communication. The structure of the brain reflects patterns of past activities of neuronal communication as much as it shapes these communications in the future which is a biological expression of the principle of progressive determination (see Section 3.2.4 on page 74).

In order to define the notion of topological structure of the whole message-passing system it is convenient to introduce the notion of time – a nominal measurement scale based on which the existence and relative frequencies of operations can be measured. In principle, an exogenously defined global time is not needed for defining a topological structure, yet is convenient to assume for didactic and reasoning purposes. Suppose, therefore, that we can define this scale in such a manner that each operation can be assigned a point in it (*timestamp*) based on the exact time it was issued. We denote such timed operation by op_{ij}^{t}, where t is the timestamp. An observer, who is observing a message passing system, arbitrarily chooses an observation period $T = (t_1, t_2)$. The **topological structure** of a system given period T is described by the frequency distribution of pairwise topological relations of all processes $op_{ij}^{t} \in OP$ of that system, where $t_1 < t < t_2$.

Having in mind these definitions, we observe the following:

i. The topological structure of a system can be measured without knowing or observing the "start" and the "end" of the system's operation, but by defining observation period T, which, of course, could span the entire lifetime of a system as a special case.

ii. The change in a system can be defined and measured by choosing the sequence of observation periods $TS = T_1, T_2, ..., T_n$, observing topological structures (i.e. distribution of frequencies of operations) for each period and then comparing them. Note that the observed change would depend on at least on two exogenously chosen parameters: the granularity of observation periods and the method of comparison of distributions.

iii. Communication channels op_{ij} can be sub-divided into sub-channels by the type of operation, allowing synchronicity, topological relations and structure to be measured per every type of operation or selected subset of operations.

iv. There is a deep similarity between defining the topological structure of the message-passing system and the notion of "near-decomposability" of hierarchical complex systems as defined by Simon (1962). Both can be expressed in terms of a matrix of frequencies (or other parameters) of interactions among elements. Hierarchy, which is a topological structure, can be inferred from properties of this matrix (see page 21).

v. Notwithstanding the above, the topological relations and history of operations of a message-passing system can be represented by a network (graph) data structure, which is the second most important component of stigmergic computing. We attend to this in more detail in the following section.

4.5.3 Graph models for data structure

A natural choice for representing the topological structure of a message-passing system is the property graph $G = (V, E, \lambda, \mu)$ (see page 42), where:

i. processes p_i are represented as vertices $v_i \in V$;

ii. communication channels between processes p_i and p_j are represented as directed edges $e_{ij} \in E$;

iii. process types and other properties are represented as labels on vertices in λ;

iv. operation types, frequencies and other properties are represented as labels on edges in μ.

The particulars of a message-passing graph are illustrated in Figures 4.11 and 4.12. An edge between two processes forms when the first process writes to an object which is afterwards read by the second process, resulting in a communication event. Formally, a **communication event** is an ordered pair of operations $op_1 = (p_i, write, X)$, $op_2 = (p_j, read, X) : op_1 \xrightarrow{\text{op}} op_2$. One or more communication events constitute a communication channel. All communication channels in a message-passing system, represented as edges between processes, constitute a message-passing graph, which can form a higher order computation process.

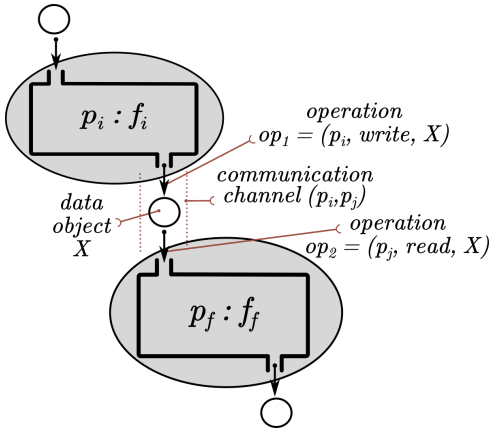


FIGURE 4.11: A message-passing channel between process p_i to p_j forms when two operations $op_1 = (p_i, write, X)$ and $op_2 = (p_j, read, X)$ are ordered by $op_1 \xrightarrow{\text{op}} op_2$. In a property graph it can be represented as a direct link e_{ij}, optionally labelled with the type of object X and timestamps of operations op_1, op_2.

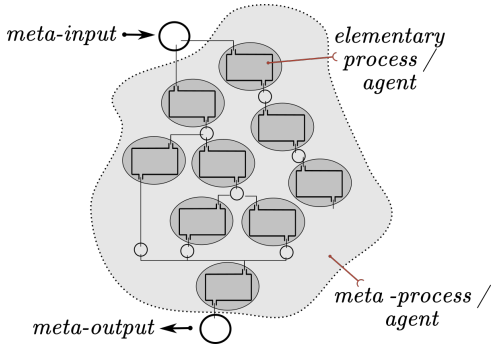


FIGURE 4.12: The message-passing graph is constructed by considering all communication channels in a system and their history of operations. This type of graph, which connects a "meta-input" object with a "meta-output" object can be considered a meta-process that can participate in more complex structures at a higher level. Note the resemblance to the population of "unorganized" Turing machines in Figure 4.6 on page 104.

As we have seen, a computation can be represented in terms of interacting processes (see Figures 4.5, 4.6 and 4.2), and therefore it is easily seen that a communication graph also represents a computing network (Gershenson, 2010) or a computation graph. Communication events and communication channels are consequences of particular orderings of operations in a system. This means that a computation graph is an emergent structure based on these orderings. We can now try to define synchronization and its built-in and emergent aspects more precisely for our purposes: (1) synchronization between two processes occurs when they establish an enduring communication channel; (2) built-in synchrony results from the interaction of processes via exogenously constructed channels by system designers; (3) emergent synchrony occurs when a processes issuing a *read* operation on a particular object *finds* or *matches* another process that has issued a *write* operation on the same object, thus forming a communication channel without defining it exogenously.

The definition of emergent synchrony is of particular importance for the computational notion of **generalized stigmergy**. First note that the distinction between direct and indirect communication can be defined using these formulations:

- a **direct communication** happens when processes issue operations into an already existing communication channel (i.e. the "writer" and "reader" of a message are known before the communication event starts);
- an **indirect communication** happens when processes issue operations without considering a particular communication channel which can (or can not) form after an arbitrary period of time since both *read* or *write* events were issued.

Generalized stigmergy in computational terms is a system of independent directly and indirectly communicating processes *with memory*, where operations or objects can be stored for an arbitrary amount of time as well as accessed by other processes. A computation happens in such a system when direct or indirect communication events form into communication channels and then computation graphs. Open-ended decentralized computing allows for dynamic adjustment of degrees of freedom of stigmergic computation in terms of relative proportion of built-in and emergent synchronization at any point in time within life of a system. Recall from Section 4.4.1 that open-ended decentralized computing effectuates computation as the processes of *integration* and *disintegration* in which heterogeneous independent processes probabilistically but reliably are assembled and reassembled to computation graphs. Guided self-organization, on the other hand (see Section 2.2.5), only approaches the integration aspect by searching for proper ways to constrain the self-organizing system in order to increase probability for it to reach *a priori* defined goal states. Obviously, open-ended decentralized computing embraces, but is not limited to, guided self-organization. Stigmergic computing is the underlying computational framework that enables open-ended computing.

Memory is the key aspect of stigmergic computing since it realizes the notion of a shared medium through which independent processes and agents indirectly coordinate their activities. In terms of the process of progressive determination, memory enables the structure of a computation graph, which influences processes and operations, as well as being influenced by them. In this sense, a computation graph, emerging from (possibly guided) self-organization of communication channels between processes, does not only realize certain global computations at every moment of the system's lifetime, but also influences its own further self-organization via progressively constraining further operations. Note that in this model the shared medium is a *distributed memory structure* composed of independent local components realized by communication channels and objects "readable" by other processes in the system. Since decentralized computing principles do not allow for any process of a system to access directly the global state of structure (i.e. states of all communication channels at any particular point in time) processes can only query states of particular local channels. Of course, as noted earlier, there is no restriction for any process, which has enough computation resources, to query each communication channel (i.e. link) in order to construct a faithful representation of the whole computation graph in its own memory. Note, however, that this can be done only via communication with other processes and can be only an approximation, since other processes will be changing the structure at the same time. Architectural aspects of such decentralized "shared medium" are discussed in detail in Chapter 5.

The next section presents and shortly discusses the "classical" example of stigmergy – *ant colony* – as a special case of generalized stigmergy using definitions developed above.

4.5.4 "Classical" example of stigmergy

Since the early 1990s, examples and mechanisms of the collective behaviour of social insects, including termites, bees and ants, have been a rich source of inspiration for scientists developing combinatorial optimization, communication networks, robotics, multi-agent systems, artificial intelligence, "swarm intelligence", decentralized computing systems and more (Abraham, Vitorino, and Grosnan, 2006; Dorigo and Stützle, 2004; Stützle and Dorigo, 1999). The explosion of this line of research was due to the realization of applicability of the mechanism of stigmergy – i.e. indirect coordination between independent actors via a shared medium (see Section 2.2.4) for computationally solving combinatorial optimization problems (Colorni, Dorigo, and Maniezzo, 1991).

Classically, the stigmergic mechanism of an ant colony optimization is expressed by the ability of individual ants (i.e. decentralized processes) to access and read the environmental signals in form of pheromones (i.e. objects) which are left by other ants (i.e. decentralized independent processes) on a terrain (i.e. global data structure). While stigmergic optimization is a powerful mechanism for achieving higher level intelligent behaviour due to coordination among agents of much simpler behaviour, it is still a simplification of general decentralized computing systems. First, observe that even social insects combine both direct and indirect communication (Bonabeau, Dorigo, and Theraulaz, 1999). Second, swarm optimization and algorithms are often constructed assuming homogeneity or at least simplicity of agents as well as a single modality of their interaction (Li et al., 2016). In more general and complex settings, especially in the case of complex agents, the computational model has to allow for hybrid direct and indirect interactions, heterogeneous agents and multiple modalities. Finally, the existence of an *a priori* defined global data structure (the terrain) is at odds with the basic principle of decentralized computing. The importance of following this principle from the point of computational efficiency and viability of a decentralized system will be discussed in Chapters 5 and 6. Let us see now how the "classical" stigmergy would be represented in terms of generalized stigmergic computing:

- First, the open-ended computing model considers the environment of any individuating agent a population of other interacting agents; therefore what is often considered the global structure of environment is represented here as a network of other agents, only with fixed topology. The environment (terrain) is a "stability landscape" which is a degree of metastability (see Section 3.2.3 and Figure 3.9). While usually, as well as in this case, the landscape is considered static and defined exogenously to computation, it need not be so in general.
- Second, agents "travelling" through such a landscape leave traces by issuing strictly local *write* operations into direct communication channels to agents that are part of the landscape. Likewise, traces are picked by processes issuing *read* operations on the direct communication channels with agents representing bits of the environment. The event of an agent representing an "ant" establishing a direct communication channel with an agent representing "a bit of local environment" implies the topological closeness of these agents, equivalent to an "ant being in a certain local point on a terrain".
- Third, agents representing "ants" may perfectly well communicate among each other via direct channels in different modalities too if that is deemed necessary.

The goal of this example is to provide a simple enough illustration of a known stigmergic process using the open-ended computational model. Obviously, the model is too rich for representing the interaction of homogeneous simple agents in a stable terrain. Our goal, however, is to establish a maximally flexible computational framework which is able to support simulation, dynamic emergence, self-organization or a representation of a broad variety of more or less complex systems and environments, including the "classical" case of stigmergy in terms of an ant colony.

4.6 Summary of the chapter

Open-ended decentralized computing is a computational model able to support the synthetic cognitive development process. Its is conceived by applying both the broad computational metaphor as well as particular methods, models and approaches developed within the domain of computer science and, particularly, decentralized computing systems research. In a manner of speaking, the open-ended computing model is the result of looking through the "computational lens" at the philosophical framework of open-ended intelligence. In that, it conceives in a different perspective, complements and expands certain aspects of the philosophical framework, rather than merely "implementing" it. Given the main tenet of open-ended intelligence philosophy of maximum freedom, the computational model allows a wide variety of configurations of decentralized systems to be potentially simulated and, most importantly, implement their self-organization and emergence via enabling the mechanism of progressive determination. Therefore, it creates the ground for simulation modelling of particular configurations as well as the emergence of decentralized systems and individuation of cognitive agents in general.

As introduced in Chapter 1, the methodological approach of this thesis is an interdisciplinary design inquiry relating the most abstract conceptual frameworks of the philosophy of individuation and becoming with concrete system implementations in software. The model of open-ended decentralized computation provides such a bridge relating theory and practice. Walking this bridge, however, requires the design of a software architecture based on the principles of computation model for performing simulation modelling experiments – i.e. "methodological blenders" (Barandiaran, 2003; Di Paolo, Rohde, and De Jaegher, 2008; Sayama, 2014). This will be approached in Chapter 5.

We started this chapter by discussing sub-symbolic and symbolic representations, symbol grounding, revisiting Simondon's philosophy of information and notions of selective and descriptive information. This leads to the precise formulation of deterministic and non-deterministic computation and the corollary that determinism is a special case of non-determinism. An overview of the contemporary and historical debate around the question "what is computation" is provided including discussion of Turing's computation models and beyond. We then propose two notions of computing: *closed computing*, based on the classical Turing (1937) model and strict interpretation of the Church-Turing thesis and *open computing*, based on the later model of Turing (1948) and a contemporary model of hypercomputation: *interactive computation*.

The second part of the chapter deals with the "conundrum of decentralization" and defines precisely the notions of centralization, decentralization and peer-to-peer

structures in terms of network science. Noting the possibility to represent computing processes in terms of graphs, we formulate the second important corollary of the chapter, that deterministic or non-deterministic properties are reflected by the structure of the computation graph of the computation in question.

Next, we discuss how an initially non-deterministic computation process can reach determinism via consensus and synchronization which allow for different levels of fuzziness. Here we observe that the process of "organizing the unorganized", proposed by Turing (1948) corresponds to the evolving structure of the computation graph. Furthermore, since the framework of open-ended intelligence and progressive determination is formulated in terms of population of independent interacting agents, different structures of the evolving computation graph reflect the notions of *preindividual*, *fluid individual* and *fully formed individual*. Recall that the scheme of synthetic cognitive development posits alternating levels of integration and disintegration of the system within a population of interacting agents. The computational expression of such a population is the message-passing framework, where notions of synchrony and asynchrony are well defined. Building on these observations and corollaries we propose the model of **open-ended decentralized computing** which conceives a population of interacting elements with dynamically changing and fluid levels of synchrony and asynchrony. The fluidity of the system is then characterized by naming possible intermediate configurations of its meta-stable landscape including built-in complete synchrony and consensus, emerging local synchrony and consensuses and complete asynchrony.

The realization of open-ended decentralized computing is conceived by generalizing the mechanism of stigmergic coordination into the notion of **stigmergic computing**. At the highest level, stigmergic computing integrates interactive computation in the message-passing framework with the fluid topology of the computation graph. The low-level semantics is developed in terms of computational primitives: (1) elementary computing *processes*, which are also *vertices* of a graph which (2) send and receive *data objects* (messages) to each other; (3) messages correspond to communication events which are persisted in the graph as *links* between vertices. Note that the integration of interactive computation and shared topology is achieved by the duality of computational primitives enabling both direct communication and communication via the shared medium. Finally, we illustrate how the "classical" case of stigmergy – ant colony – can be expressed within the generalized model of stigmergic computing.

With the model of open-ended decentralized computing we show how the mechanism of progressive determination, central to the notion of open-ended intelligence and synthetic cognitive development, computationally can be expressed in terms of synchronization of communications in an asynchronous computing system composed from two levels – message-passing framework and fluid graph structure. Utilizing semantics of stigmergic computing, open-ended decentralized computing effectuates computation as the processes of integration and disintegration in which heterogeneous independent processes probabilistically but reliably are assembled and reassembled to persistent, yet fluid computational graphs.

The best way to understand the relation between stigmergic computing and open-ended decentralized computing is to consider the latter a higher level model, while the former as the basis on which this higher level model can be expressed. Nevertheless, both concepts are in close relation and to a large degree overlap yet are explained separately, largely for didactic reasons. Furthermore, both concepts

are very general models of computation which do not try to limit the range of simulation models and dynamics that can be expressed with them – just as the universal Turing machine does not limit the range of algorithms that can be implemented with it.

In the next chapter we will propose the software architecture for the actual implementation of open-ended decentralized computing.

Chapter 5

Towards architecture for open-ended decentralized computing

This chapter proposes an implementable software architecture for realizing the computational model developed in Chapter 4. The main tenets of the computational model and, consequently, the functional requirements for the architecture are:

i. *Population of processes*: asynchronous operation of large numbers of independent and heterogeneous elementary computational processes encapsulated into agents (see Section 4.5.1);

ii. *Custom behaviours*: ability to define and implement arbitrary complex computational processes in the Turing (1937) sense within every individual agent of the system;

iii. *Heterogeneity*: efficient distribution of computational resources for each agent as required by their processes, which can differ widely in logic and computational complexity as estimated by the time and memory needed for process execution;

iv. *Direct interaction via message passing*: ability of agents to interact continuously with each other on a peer-to-peer basis, as required by their own behaviours. For this end, agents have to be encapsulated into a message-passing framework (see Section 4.5.2);

v. *Indirect interaction via shared medium*: a decentralized memory in terms of graph data structure, readable and writeable by each agent of the population in a concurrent and independent way (see Section 4.5.3) needed to achieve *stigmergic computing* (see Section 4.5).

vi. *Evolving neighbourhood structure*: each agent is embedded into the graph data structure via its explicit relations (links) to one or more other agents; links can be dynamically changed during operation (created, deleted, arbitrary properties changed);

vii. *Radical decentralization*: an agent and its processes can interact with the shared medium only via its local structure – i.e. explicit relations to the neighbours;

In terms of software design, these requirements can be satisfied by tightly coupling two components: *actor framework* and *graph computing engine*. The actor framework enables requirements *i* to *iv*: population of processes, custom behaviours, heterogeneity and direct interaction via message passing. The graph computing engine enables requirements *v* to *vii*: indirect interaction via shared medium, evolving neighbourhood structure and radical decentralization. These two components and their interaction determine the software architecture of open-ended distributed computing and are discussed in detail in the following sections.

5.1 Actor model and framework

5.1.1 Description of the model

The **actor model** is a mathematical model of concurrent computation, proposed by Hewitt, Bishop, and Steiger (1973) and further developed by Clinger (1981), Greif (1975), Agha (1986) and others. It is conceptually based on one kind of object – an autonomous communicating actor (processor, agent, etc.) which does not presuppose any representation of primitive data or control structures. Data structures can be programmed or hard-wired and encapsulated or dynamically evolved to each actor separately or to an ensemble. This includes possibilities of emergent computation, genetic or evolutionary programming and other local computational mechanisms. Actors communicate via immutable messages and in this sense constitute a message-passing framework. Patterns of message passing between actors is the only control structure which defines the functionality and behaviour of a particular actor system as a whole (Hewitt, 1976). Note immediately that no constraints on message-passing patterns or even requirements for their existence are defined by the lowest level of the actor model. System engineers enforce these constraints via high level design or let them emerge during computation.

An **actor** is a self-contained, interacting, independent component having a *behaviour* and a *mail address*. An actor's behaviour is expressed by a computational *process* having inputs, outputs (see Figure 4.10) and computational resource requirements, including *storage*. Actors communicate with each other via asynchronous message passing using their mail addresses. Basic actor primitives, which allow all further abstractions to be built, are: (1) *send(a,v)*, which creates a message with contents v and sends it to an actor with address a; (2) *newactor(e)* – creates a new actor evaluating expression e and returns its mail address; (3) *ready(b)* – frees the actor to accept the next message and readies it to execute a behaviour b. Note that all computations in the actor model are built by these primitives only, which are necessarily executed on behalf of an actor in an actor system. No primitive can be executed without an actor that executes it. For example, *newactor(e)* can be executed only by another actor, in which case they may establish a parent-child relationship within the topology of the actor system. This primitive allows new processes and data structures to be created dynamically. The *ready* primitive gives actors a history-dependent behaviour by defining and delineating groups of actions as atomic. The *send* primitive is an asynchronous analogue of function application which causes an action on behalf of another actor by putting a message in its mailbox (Agha and Jamali, 1999).

Actor system is a container which enables actor creation, addressing and message passing, but is otherwise computationally passive and ignorant of internal processes encapsulated within actors. Actors can know the mail addresses of other actors, which makes them *acquaintances*, or *neighbours*. Mail addresses can be contained in messages, which allows for a dynamic actor interconnection topology, where actors can connect directly to each other based on indirectly passed and received messages (Callsen and Agha, 1994; Houck and Agha, 1992). The actor system therefore is a dynamic network (graph) structure, where nodes are actors with links to their mutual acquaintances. Using these links, every actor can in principle reach every other actor and learn local or global network structure via distributed message-passing algorithms (Raynal, 2013).

Flexible actor creation and reconfiguration supports the incremental construction of distributed computing systems; using the three primitives, actor systems can be dynamically configured and reconfigured. New actors can be created, destructed and connections between them established or broken as computation proceeds. Therefore, the actor model does not require the shape of the computational problem to be completely established or execution resources to be fixed before computation is initiated (Agha and Jamali, 1999). These properties clearly position the actor model and semantics in line with the concept of open computing (see Section 4.2.5).

Further, instantaneous snapshots of an actor system manifest particular configurations of its message-passing graph (ibid.). Recall from Section 4.5.3 that a partial order of messages in a *communication graph* define the *computation graph* and, therefore, the logics of the particular computation. Therefore, the actor system as a whole can be seen as a meta-computation built or emerging from interactions among lower level computational processes, which is the major requirement of the open-ended decentralized computation model.

Clinger (1981) proposes a convenient formalization by positing *event* as a primitive object of the model along with *actor*. An actor represents a computational agent that can be thought of as having its own separate processor on which to run. An event then represents the arrival of a message at the target actor. This model uses partial orders of events to represent concurrency analogously to the message-passing aspect of stigmergic computing (see Section 4.5.2). Formally then, a computation of an actor model is a structure $\langle E, A, T, \xrightarrow{\text{act}}, Arr \rangle$, where:

i. E is a set of events;

ii. A is a set of actors;

iii. T is an association of each event with the target actor $T : E \to A$;

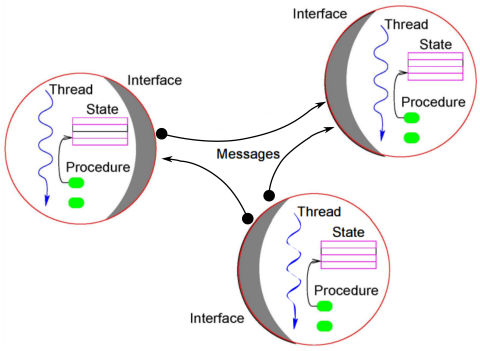
iv. Arr is a collection of irreflexive total orderings $\xrightarrow{arr_a}$ of message arrival events E, defined on $T^{-1}(a); a \in A$;

v. $\xrightarrow{act}$ is the irreflexive partial ordering of events E, such that no event has more than one immediate predecessor.

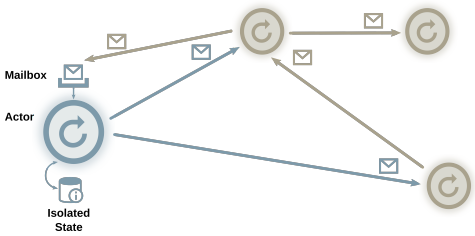
Observe that the last two items, i.e. orderings $\xrightarrow{act}$ and $\xrightarrow{arr_a}$ define a message-passing framework with only direct messages, where each message has an activation (at the source actor) and an arrival (at the target actor). Events that define computation are arrivals; activations *must* happen for each arrival to happen, but their order is not important for computation. If we assume the global time in an actor system, both arrival and activation of an event e can be assigned respective timestamps ts^e_{act}

and ts^e_{arr} that would define the amount of time it takes for a message to reach its target: $time^e = ts^e_{arr} - ts^e_{act}$. It is a fundamental property of the actor model that this time is finite, but unbounded – that is, messages are guaranteed to arrive at their targets, but there is no upper limit to the amount of time $time^e$ that this event can take. This property allows the actor model to express non-deterministic computations without reference to the halting problem which makes it "super-Turing" (Hewitt, 2010, 2013).

The **actor framework** is an actual software framework, package or system which enables development of actor systems. An actor framework is normally tied to the programming language in which it is implemented, or, at least, which it supports. At the time of writing, many popular programming languages had an actor framework, including C++, languages based on Java Virtual Machine and more[1].



(A) Actors encapsulate a thread and state. The interface is comprised of public methods which operate on the state. Adapted from Agha and Jamali (1999).



(B) An example network of several actors. Each actor has its own mailbox and isolated state. Based on its designated behaviour, the actor responds to incoming messages by sending new messages, spawning new actors or changing its future behaviour. Adapted from Erb (2012).

FIGURE 5.1: Actor model and system

Hewitt (2017) notes that the fundamental advance of the actor model was the decoupling of a sender from the communications it sends – once a message has been sent, it loses any relation to the sender and is delivered on a best effort basis. This is a sharp difference from other models of concurrent computation, where message sending is tightly coupled to the sender, which synchronously transfers the message to some persistent structure – e.g. buffer, queue, mailbox, channel, broker or server – for temporary storage and retrieval by a reader. This absence of the global state in the actor model allows it to embrace inconsistencies inherent in large decentralized systems. Actually, consistency and synchronous execution are special cases within the actor model and have to be explicitly enforced if considered necessary by designers. Therefore, the model implements determinism as a special case of non-determinism, which is at the very basis of open-ended decentralized computing (see Section 4.2.3).

The absence of the requirement of persistent global state and synchronous execution, while allowing for richer and more scalable models of computation, is also difficult to reason about and creates a great deal of confusion, which has possibly contributed to complications in the adoption of the actor model in practical computing and software development. At the time of its conception and early development

[1]For an exhaustive list, see https://en.wikipedia.org/wiki/Actor_model.

in the 1970s, the actor model did not enjoy the popularity of the other computational models of Neumann (1945) and Turing (1937), also due to the limitations of its physical implementation on existing computer architectures. Yet the model gained popularity with the development of multi-core computer architectures, parallel and distributed computing, where it fits and excels naturally (Greif, 1975). Observe also that the actor model effectively implements the concept of interactive computation (see Section 4.2.5) and constitutes a single currently viable super-Turing computing model (Hewitt, 2013; Nayebi, 2014).

5.1.2 The fundamental principle of (de)centralized computing

The actor model was originally introduced as a "universal modular formalism for artificial intelligence" that does not require a global observer, algorithm or manager. It aimed to model intelligence [...] in terms of a society of communicating knowledge-based problem-solving experts. In turn each of the experts can be viewed as a society that can be further decomposed in the same way until the primitive actors of the system are reached". The nature of the communication mechanisms needed for effective problem solving is modelled "by a society of experts and the conventions of discourse that make this possible" and is aimed at developing a framework for problem solving involving parallel versus serial processing and decentralization versus centralization of control and information storage (Agha, 1986; Hewitt, 1976; Hewitt, Bishop, and Steiger, 1973).

Hewitt and Manning (1996) proposed a notion of *participatory semantics* in order to account for the research of multi-agent systems in terms of scalable, plural, open information infrastructures, comprised of humans, equipment and services (see Figure 5.2). It can be seen as a generalization of the mathematical actor model for concurrent computation to reason about any participatory, distributed and open multi-agent activity, including human, social, biological and especially hybrid socio-technological systems.

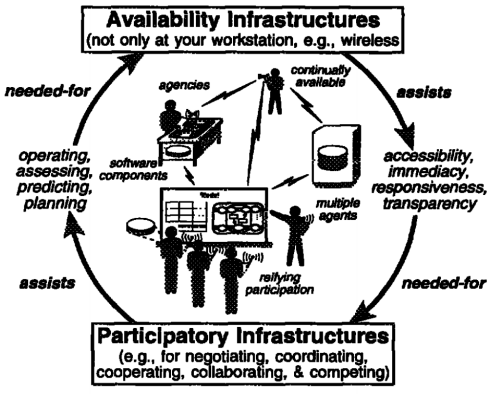


FIGURE 5.2: Participatory semantics: participation in multi-agent systems depends on a synergy of interdependent, overlapping and mutually supporting components: availability infrastructures and participatory infrastructures. Adapted from Hewitt and Manning (1996).

Participatory semantics and the actor model are inspired by concepts in physics,

such as (quantum) indeterminacy and relativity (Hewitt, 2010; Hewitt, 2006). Concretely, it relies on the concept of information that is necessarily incomplete and relational. Information expresses the correlation of configurations of interacting physical (or computational) systems – the principle that Hewitt (2006) derives from relational quantum mechanics. This principle states that "the way distinct physical systems affect each other when they interact (and not of the way physical systems 'are') exhausts all that can be said about the physical world. The physical world is thus seen as a net of interacting components, where there is no meaning to the state of an isolated system". This principle is in line with the Simondonian concept of information (see Section 3.2.1) where information can be signified only in the context of the interaction but not exogenously. In terms of Simondon's philosophy of information, the meaning of a state of a system S_1 can only be signified by an observing system S_2 for which the state of S_1 is significant, or meaningful. Concepts of quantum mechanics and philosophy of information force one to give up notions of a description of a system independent from the observer and absolute state (ibid.). The actor model expresses these concepts by excluding the notion of a global state of the system from the definition of computation and discouraging its use, while leaving the ability to enforce it through software design decisions – i.e. any actor, given enough resources and time, can in principle learn a global state.

The issue of the global state of the multi-agent system and an agent's ability to acquire it via learning enables the computational metaphor to be used more precisely for addressing "the conundrum of decentralization" (see Section 4.3). Consider an actor system where each actor knows and can send a message to a certain random number of other actors, which is strictly less than the number of all actors in the system. This defines an *a priori* decentralized system. First, observe that such an actor system can be represented in network or graph form. Second, in order for any actor to have knowledge of the global state of the system, it has (1) to know all other actors and (2) to have a local representation of each actor's state in its own memory. Consider then a simplified graph structure $G = (V, E, \mu)$, with actors $a \in V$ and their "knows" relations $e^{\mu}_{i,j} \in E$; $\mu = knows$; $e : a_i \xrightarrow{\text{knows}} a_j$. If one of the actors in such a system – let us denote it a_{omni} – is set to learn the global structure of the graph, it needs: (1) to connect to all other agents with edges $e^{\mu}_{omni,j} \in E$; $j \in [1, |E|]$, (2) enough memory to store faithful representations of all actors $a \in V$; $a \neq a_{omni}$ and (3) enough memory to store faithful representations of all links between all other actors $e^{\mu}_{i,j} \in E$; $i, j \neq omni$.

We can now infer from the above two important considerations. The first is that a_{omni} has to have at least as much memory storage as the whole actor system in order to represent it precisely. Considering decentralized systems of a massive scale (e.g. internet), this is a formidable practical constraint. The second consideration is more subtle and has to do with the dynamic nature of multi-agent systems. Suppose that a_{omni} realizes that, instead of holding a copy of each actor's state in its memory, it could simply retrieve them at demand through existing connections – using the network structure as a kind of "external memory". This way, a_{omni} could have access to the global state at will via communication: i.e. by broadcasting a single message ("send me a copy of yourself") to all actors and receiving one message from each of them. This would save a_{omni} from prohibitive memory requirements. But, from a computer-scientific perspective, this is just trading a certain kind of computational resource for another – in this case memory for time, since message sending and receival necessarily takes time and processing (Li et al., 2018). Furthermore, the

context of the dynamic decentralized multi-agent system means that actors may and do change their states constantly, and therefore a_{omni} has to send and receive *all* messages fast enough so that no actors in the network change before all answers come back. Again, considering actor systems of massive scale and their asynchronicity, such a scenario is close to impossible. Actually, even having a memory of the scale of the whole network does not save a_{omni} from the need to update its copy every time the network changes, which implies constant communication over existing links.

Based on this discussion we can formulate the fundamental principle of decentralized computing. Suppose that a_{omni} has a computational task that needs information from each actor in the network as an input and can choose an algorithm for any possible combination of *storage* and *communication*, as explained above. The actual choice of the combination would depend on the following parameters:

i. cost of internal storage $cost_{memory}$;

ii. cost of communication $cost_{comm}$;

iii. speed of communication in terms of the amount of time needed to exchange messages with each actor t_m and

iv. rate of change in the actor system TS, defined as the maximum time between two observations which do not differ from each other (see Section 4.5.2).

The actual choice of parameter combination involves a trade-off which depends on the immediate context of a computation. Actor a_{omni} would be inclined to use more internal storage when storage costs $cost_{memory}$ decrease, communication costs $cost_{comm}$ increase, speed t_m decreases and rate of change decreases, i.e. TS increases. Likewise, it would be inclined to use more communication over existing network connections when storage costs $cost_{memory}$ are high, communication cost $cost_{comm}$ is relatively small, the speed t_m is high and rate of change is high, i.e. TS is small. This principle emphasizes that decentralized computing systems are inherently relativistic – something that the actor model embraces at the conceptual level. Furthermore, this principle is closely related to the fundamental trade-off between the comprehensibility of an environment and the information loss about the "true" nature of it at each moment of comprehension in a generally intelligent system, embodied and embedded in certain environments and subject to changing environmental situations, as expounded by the notion of *ecological rationality* (see Section 2.2.3).

5.1.3 Extending the actor model with mobility and location

Another key issue in developing multi-agent systems that the actor model allows us to address at a fundamental level is *mobility*. Obviously, implementation of the model requires computing resources for enabling processing, storage and communication for processes which actors encapsulate. The physical implementations of such computing resources are by convention called *nodes* – i.e. servers, processors, computing centres, etc. – that together make a distributed computing infrastructure of cloud infrastructures, sub-networks and eventually a global network. Mobility allows an agent to migrate freely from one such node in a distributed computing environment to another to seek "better" execution environments (Agha and Jamali, 1999). It allows actors to be consolidated into physical assemblages (in a very similar sense discussed in Section 3.2.2) in order to enable a particular computation graph in efficient manner. For example, if an agent needs to access a large amount

of data (e.g. machine learning dataset) in different locations, it could make sense for an actor to migrate to these locations and do the processing there. This flexibility can be approached from at least two aspects – a computer-scientific and economic – both stretching the limits of computational systems as usually understood. The full discussion of these aspects is outside the scope of this work, but it is important to mention them for suggesting the significance and possibilities of the actor model for multi-agent systems development.

First, fundamentally, computational complexity is expressed as a combination of the memory (space) and time needed for a particular computation to perform. The capacity of physical computing resources is expressible by the same combination. The whole field of computer science, on both theoretical and practical levels, essentially deals with only one problem – finding and enabling efficient combinations of memory and time for particular computations. In distributed systems time has two constituents: processing time and communication time (note that all non-trivial computing systems are distributed in this sense – including a single processor). Tight assemblage of all processing and storage units in terms of physical locality allows the same computation to be performed more efficiently by reducing the time of communication. Mobility in the actor model allows the capability to be conceived of computing systems to search and find such combinations themselves.

Second, in order to support a system in which actors would autonomously yet collectively search for assemblages leading to "better" configurations, as required by different tasks and problems, the notion of *computational economy* is needed. Such an economy would define the costs of physical computing resources and data transfer that actors would need to take into account when forming assemblages – or larger computational units. Moreover, in order to describe such an economy, notions of actors' location, mobility and execution context should be defined in a way that could be accessed and reasoned by the actors themselves. This requires an extension of the otherwise location-transparent actor model with a global yet dynamic *topological structure* that can be accessed and modified by actors. Recall that a topological structure is a necessary component of the open-ended decentralized computing model, allowing the evolution and self-organization of message-passing systems (see Section 4.5.3). Here we discuss an economic extension of the actor model proposed by Agha and Jamali (1999) which makes use of the topological structure and greatly enhances the fluidity of the model, taking into account the pragmatics of real-world computing and distributed systems. The proposed extension defines the following important additional components:

i. *Naming scheme*. Since actors can migrate, their addresses should include location information in the form of $h.a$, where a is the globally unique identifier and h is the node at which it resides[2]. This makes the naming scheme in the actor model somewhat similar to the Universal Resource Identifier (Berners-Lee, Fielding, and Masinter, 2005). Furthermore, the naming scheme should support referencing ensembles or groups of actors in a pattern-based way in order to be able to access new services and to know when old services become unavailable (see Figure 5.3a). This enables roles to be defined for agents and to be addresssed, rather than naming individual agents which, in a dynamic system, may migrate, multiply or get destroyed.

ii. *Economic mechanism* deals with the resource allocation in a multi-agent system

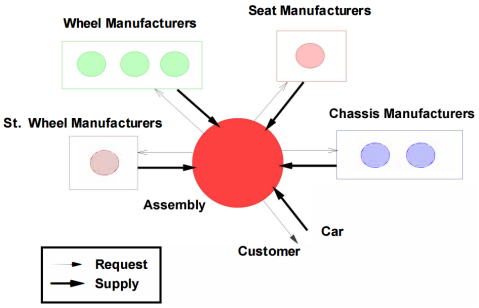
[2] Akka actor framework (http://akka.io used for software implementation described in Chapter 6)

where agents and nodes are mobile, independent and have a certain notion of "choice" in their actions. On behalf of an agent it would involve a selection of computational tasks to perform; on behalf of a node, selections of agents which are allowed to use its resources. For this end, the notion of the *global currency unit (GCU)* is used in order to establish a basis for comparison between different offers for computational resources on behalf of nodes and demands for different computational tasks on behalf of agents. Therefore, the *semantics of resource bounded agents* is proposed, which constitutes the basis of the economic mechanism, where $T_{res(a,h)}$ is a function that represents a contract between an agent and a node hosting it and determines the cost of computation (whether in GCUs or otherwise). Note that such a contract is a result of negotiations between agent and a node, considering that costs of computational capacities and demands can vary over time.

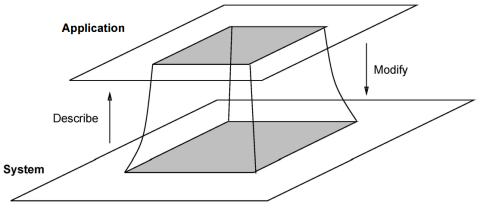
iii. *Customizable execution contexts* involves a technique of *computational reflection* (Figure 5.3b) which enables individual actors to have a continuous interaction with their environment to determine available resources, relate to their own state, search for and request additional resources and by that provide evolving resource consumption strategies (Venkatasubramanian and Talcott, 1995). These are needed for an agent to specify requirements for negotiation with a hosting node, besides the bare estimation of computational resource requirements – e.g. programming languages, library dependencies, hardware components, etc.

iv. *Coordination* and *interaction protocols*. The power of the actor model comes from the exploitation of parallelism, distribution and mobility in ensembles of agents. It promises orders of magnitude greater computational abilities than that of individual actors, which are nothing more than conventional sequential programs. As each problem-solving agent possesses only an incomplete local view of the system and limited computational power, it must coordinate with other agents in order to achieve coherent and globally viable solutions. Since in the actor model there is no global coordinating actor, it has to happen from the bottom-up via synchronization, coherence and functional coordination – the process of synthetic cognitive development. In order to provide a minimal basis for coordination in a computational medium such as an actor system, interaction protocols and policies have to be in place (see Figure 5.3c). Apart from functional coordination, interaction policies enable the system to deal with failures, which is the norm rather than the exception in distributed and decentralized systems.

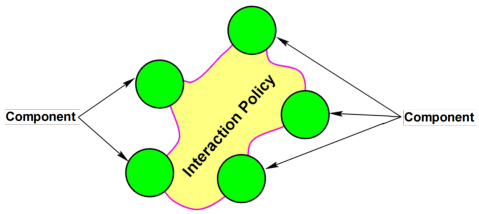
These extensions of an actor model are components of what was called *decentralized IT governance* in Box 4.1 and are attempts to address a key challenge in multi-agent systems and distributed artificial intelligence research and practice – the ability to coordinate decentralized agent ensembles. Apart from actor mobility and economy, fluid topology is most importantly required for computational implementation of concepts of progressive determination (Section 3.2.4), synthetic cognitive development (Section 3.3.3) and stigmergic computing (Section 4.5). As propounded by the model of open-ended decentralized computing, the fluid topology is provided by integrating the message-passing framework of the actor model to graph computing.



(A) A car assembly factory. The assembly sends requests to actor spaces, whose membership may dynamically change.



(B) Computational reflection – actors have formal representation of their own applications; systems (nodes) have formal representation of their computational resources and contexts; based on this they can establish a communication.



(C) A distributed system consists of a set of components carrying out local computations and interacting in accordance with a set of policies.

FIGURE 5.3: Extension of the actor model with resource model and economy. Adapted from Agha and Jamali (1999).

5.2 Graph computing

Graph computing is a quickly developing new paradigm of computing which leverages network science and graph theory (see Section 2.2.4) to represent data, programs and their interactions in distributed computing environments. It is a large set of technologies and a way of modelling the world in terms of entities (nodes) and their relations (semantic or analogous associations, causality, information and control flows and more). The graph computing field is very diverse and evolves quickly by featuring new and augmented ways to computing on theoretical and practical levels. In this section we introduce and discuss several aspects of graph computing directly relevant to the implementation of open-ended decentralized computation model.

5.2.1 Graph databases

The first large scale application of graph computing technologies appeared in the data management field in the form of *graph databases*. This is due to many interrelated reasons, but perhaps most importantly to the enormous global economic impact of data management technologies in today's society. Contemporary commercial graph databases[3], utilizing an idea that real world systems and data about them can be represented in a graph form (Rodriguez, 2017), was conceived in 1999 as a solution to the difficulties of legacy data management technologies for coping with the increasing amounts, volatility and analytic requirements of the IT industry (Robinson, Weber, and Eifrem, 2015). By 2017, many large data management technology

[3]Neo4j, https://neo4j.com/

and infrastructure providers[4] had added graph database offerings to their portfolio, resulting in mainstream adoption. The field has a huge potential for growth, considering the ever growing demands for scale, speed, data analytics and security of increasingly decentralized global information systems. It turns out that graph database technology excels in at least three areas largely affected by the trends of industry requirements: exploiting information encoded in relations between data points, managing inconsistent data and dealing with high velocity and dynamics.

A good image for discussing graph databases and computing is the Semantic web (Berners-Lee, Hendler, and Lassila, 2001; Easley and Kleinberg, 2010; Shadbolt, Berners-Lee, and Hall, 2006), which consists of documents, servers, web sites, programs and data points related in a disorderly way among themselves via hyperlinks. It is really an integrated data structure, albeit decentralized on a huge scale, without a single entry point, allowing every part to be changed and connected or disconnected at will without a mechanism that would enforce its global consistency. First, observe that it is a largely irregular graph structure – a "giant global graph" (Berners-Lee, 2007). Second, it is based on a family of well-defined graph data models, such as the resource description framework (RDF) and querying languages, such as SPARQL (Cyganiak et al., 2014; Harris, Seaborne, and Prud'hommeaux, 2013). In order to provide the ability to execute SPARQL queries on the RDF modelled data structure(s), there needs to be a server with persistent storage of RDF triples and an interface for accessing it – which is a type of graph database[5]. Another data model for representing linked data, developed almost at the same time as RDF, is the labelled property graph (introduced in Section 2.2.4). Unlike RDF, the goal of the property graph was not so much to exchange or publish data, but rather about efficient storage and fast querying (Barrasa, 2016). Historically therefore, these two graph data models took somewhat different paths of development which are still conceptually close (e.g. almost all property graph-based graph databases can store RDF and allow SPARAL querying)[6].

Graph databases are conceptually very different from the data management technologies which reigned in the 20$^{\text{th}}$ century – relational database management systems (RDBMS). It is worth articulating this difference more concretely in order to appreciate the paradigm shift. RDBMS provide so called *ACID* guarantees:

i. *atomicity* – all operations in a transaction must succeed or every operation is rolled back;

ii. *consistency* – transactions cannot violate declared system integrity constraints;

iii. *isolation* – concurrent transactions must produce independent results;

iv. *durability* – results of applying a transaction are permanent, even in the presence of failures.

Graph databases, as well as other NoSQL databases (Madison et al., 2015), provide much less strict *BASE* guarantees:

i. *basic availability* – the datastore appears to work most of the time;

[4](1) IBM Graph; (2) Microsof Azure Cosmos DB; (3) Oracle Spacial and Graph; (4) Amazon Neptune; (5) Datastax Enterprise Graph; (6) and many more.

[5]See W3C wiki page listing "large tripple stores for a non-exhaustive list.

[6]Lately there are attempts to bridge the two graph data models and databases research – see the announcement of W3C Workshop on web Standardization for Graph Data in March 2019.

ii. *soft-state* – they do not have to be write-consistent nor do replicas have to be consistent all the time;

iii. *eventual consistency* – datastores exhibit consistency at some (later) point.

Clearly, *BASE* requirements are much softer than *ACID* and that allows effective handling of the *CAP theorem*[7] of distributed data management, which states that when data is not localized in one server, one has to choose between availability and consistency because both are not available at the same time (Robinson, Weber, and Eifrem, 2015). Note that this does not prevent system developers from enforcing stricter properties via high level software design – something that is often needed for business logic.

Related to the image of the semantic web as a graph database in the above sense, consider the global brain (see Section 2.1.3), a human brain or a neural-network-like implementation of artificial general intelligence from the same perspective[8]. All these systems hold dynamic data in different forms, which is globally inconsistent and most probably not even eventually consistent, yet are uninterruptedly used (i.e. queried) for reasoning and decision making in concrete and time-critical situations. Inconsistency of a data structure means that two perfectly identical queries may produce different results. At least partially due to this, the discussion about graph computing is shallow without considering the languages and mechanisms of querying data – *graph traversals*.

5.2.2 Graph traversals

Graph traversal is a systematic method of exploring vertices and edges in a graph. Traversals are represented using special purpose *graph traversal languages* which formalize an abstract description of a legal path through a graph. Graph traversing is then a process of visiting (checking, updating or modifying) vertices and links of a graph, based on the *imperatively* defined constraints in this language by a user (Rodriguez, 2008a). It is different from the notion of *graph pattern matching* which is another form of querying a graph database with *declarative* statements. This is an important difference. Graph pattern matching, in line with the long established data querying model, returns a result set that binds variables to values in a database as constrained by the logic of the query. Examples of such languages are SQL[9], SPARQL[10] and Cypher[11]. In graph traversal model, the logic of query is encapsulated into a program (*traverser*) which then walks the graph according to the formulated constraints. The result of a traversal are graph locations (vertexes and edges),

[7] CAP theorem, formulated by Eric Brewer, states that web services (which are the special case of a decentralized system) cannot simultaneously ensure all three properties: **consistency** (i.e. the client perceives that a set of operations has occurred all at once), **availability** (i.e. every operation must terminate in an intended response) and **partition tolerance** (i.e. operations will complete, even if individual components are unavailable) (Pritchett, 2008).

[8] It can be argued that data management technology relates only to symbolic data and does not do any good when dealing with sub-symbolic representations, which arguably are needed for neural networks. However, it has been shown that symbolic representation of a labelled property graph can be reduced to an undirected graph without losing information (Rodriguez, 2008b) and that the precise meaning of every link in a graph depends on its neighbouring linking patterns (Rodriguez, Pepe, and Shinavier, 2010).

[9] https://en.wikipedia.org/wiki/SQL

[10] https://en.wikipedia.org/wiki/SPARQL

[11] https://neo4j.com/docs/cypher-manual/current/

their properties and any data structures aggregated by visiting them. Gremlin[12] graph traversal language supports both graph pattern matching and graph traversal models, but the latter is its most important property and distinguishing feature. Furthermore, apart from the ability to formalize graph traversals, Gremlin is also a virtual machine, distributed execution model and a programming language that can be leveraged in general-purpose programming (Rodriguez, 2015b; Rodriguez and Bollen, 2007). Concretely, it can be:

i. embedded in any host programming language, with currently available drivers for Java, Python, JavaScript and others[13];
ii. extended by users wishing to leverage the terminology of their usage domain[14];
iii. optimized via an extensible set of compile-time rewrite rules[15];
iv. executed within a multi-machine compute cluster or a parallel architecture in distributed manner;
v. evaluated using a variety of graph traversal models, including depth-first, breath-first or hybrid ordering;
vi. represented within the graph itself via the theoretical existence of the Universal Gremlin Machine, effectively erasing the distinction between data and program, since both can be represented in the same structure.

Gremlin is therefore both a major technology of the graph-computing landscape – which is still somewhat "under the radar" (Russell, 2019) – and an actualization of the new and still consolidating branch of computer science – *graph computing theory*[16]. The conceptual aspects of graph computing, as represented by Gremlin, are of particular interest with respect to the open-ended decentralized computing model. It goes well beyond database management systems and graph databases, which can be considered "only" the first commercial application of graph computing.

Let us return to the example of the Semantic web and the internet at large to illustrate relevant conceptual aspects of graph computing. First, note that while the notion of the web as a graph still evokes an image of the static structure of nodes – web pages hosted on locationally stable servers – linked with edges (Kleinberg et al., 1999), the internet as a whole has long evolved towards the dynamic network of interacting processes: web browsers as virtual machines running asynchronous JavaScript programs, mobile applications and cyber-physical systems interacting via web APIs[17] and the like. Furthermore, currently emerging technologies of the decentralized web, such as IPFS[18], the Safe Network[19], *fog* and *edge* computing (Matt, 2018), blockchain (Box 4.1) based decentralized smart contract platforms[20] and many more, promise to altogether decouple content, data and executable code from any physical location, server or client implementation. Such a dynamic network of interacting processes is best conceived as a decentralized computing infrastructure in

[12]https://tinkerpop.apache.org/gremlin.html
[13]https://tinkerpop.apache.org/query-languages.
[14]https://www.datastax.com/dev/blog/developing-a-domain-specific-language-in-gremlin.
[15]http://tinkerpop.apache.org/docs/current/reference/traversalstrategy.
[16]M. Rodriguez, personal communication, December 5, 2016 and Anadiotis (2018)
[17]https://en.wikipedia.org/wiki/web_API
[18]InterPlanetary File System (IPFS)
[19]https://safenetwork.tech/
[20]Projects like Ethereum, EOS, Holochain, NEO develop systems which allow loosely and dynamically networked computers to become one virtual computing machine without single point of access or failure.

terms of interactive computation (see Section 4.2.5), actor model (Section 5.1) and an ecosystem of emergent computation graphs (see Figure 4.5). With this image we turn to selected conceptual aspects of graph computing that are instrumental for the architecture of open-ended decentralized computing.

5.2.3 Vertex-centric spreading activation

When a person, program or a mobile application "queries" the web for information, one does not have access to all information residing in all other nodes – servers, web pages, programs or mobile applications – at once. One does not even know, or need to know, how many other nodes exist in the network. Instead, one accesses the network via its local node and follows from link to node to another link selectively and iteratively until reaching the desired node or content. This is a primitive example of vertex-centric graph traversal where each traversal has to start from an existing node in the graph and traverse through its neighbouring links. This operation can be written formally in imperative graph traversal language. The traverser – a program encapsulating logic of the traversal – walks through nodes and links, checks their properties and dynamically decides its next step. This decision is dictated by its own logic, local information of the node or edge it is on, and *path history* – information collected during the traversal (McCune, Weninger, and Madey, 2015; Rodriguez, 2008a). An example of an actual Gremlin traversal is depicted in Figure 5.4.

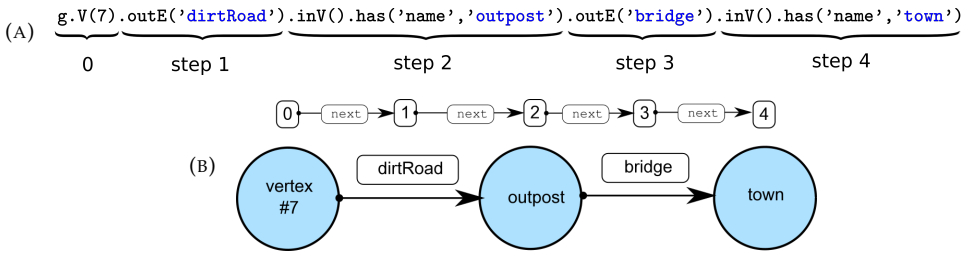


FIGURE 5.4: Graph traversal as an expression 5.4a in Gremlin and as a graph structure 5.4b. Both describe a legal path in a road map graph (Figure 5.5). 5.4b is a compact simplified representation. For a fully executable graph representation of a traversal see Rodriguez (2015b).

The expression of Figure 5.4 instructs that the traversal starts at vertex $V(7)$ and performs four steps, if such a legal path exists in the graph: (1) travel via edges *dirtRoad* (2) to all vertices named *outpost*, (3) then travel via edges *bridge* (4) to vertices named *town*. If a legal path exists in the graph that matches this description, that is, if a traverser can actually walk through such a path when landed on the starting vertex $V(7)$ of the graph depicted in Figure 5.5, then the result returned by it are vertices of the graph that represent all reached *towns*. Note again that the actual result of the operation depends not only on the logic of expression, but also on the structure of the graph that is being traversed.

When applied to the graph representing a simple road map, depicted in Figure 5.5, the traversal of Figure 5.4 returns three vertices: $V(65)$, $V(66)$ and $V(67)$, which represent towns in the graph and the *halting* condition of the traversal expression (i.e. last step). When faced with multiple legal paths (roads) in the graph, traversers *spread*, or branch, and walk all possible legal paths simultaneously. This

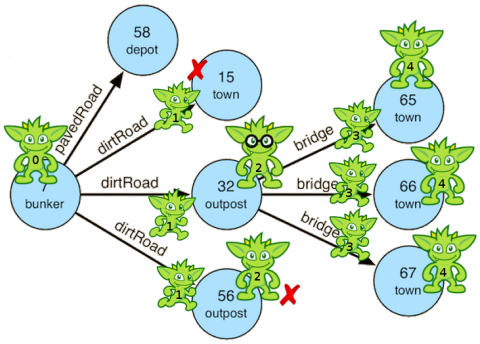


FIGURE 5.5: Graphical depiction of vertex-centric spreading activation of the Gremlin expression of Figure 5.4. Numbers on traversers correspond to the Gremlin expression steps of Figure 5.4a. Adapted from Rodriguez (2015a) with changes.

constitutes a vertex-centric, semantically constrained spreading activation (see Section 2.3.1) through the graph. More formally, a traversal expresses a path algebra expression, according to which a traverser walks through the graph structure and returns all encountered legal paths (Rodriguez, 2008a; Rodriguez and Neubauer, 2011; Rodriguez and Shinavier, 2010).

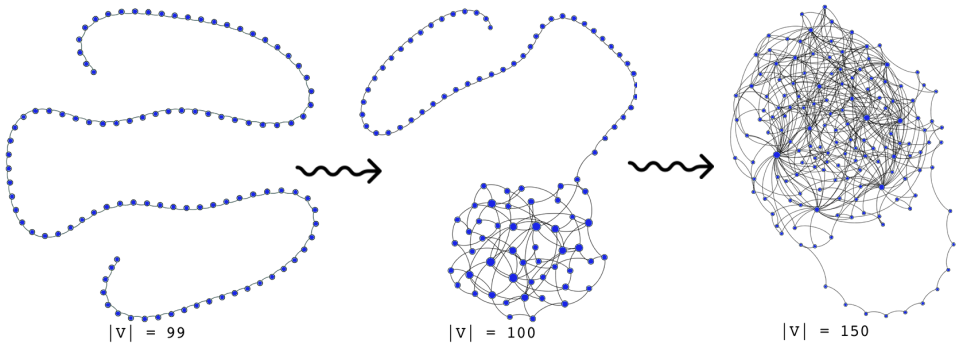
Importantly, traversers can mutate a graph when walking it – i.e. change properties (e.g. *dirtRoad* to *pavedRoad*), create new vertices (e.g. *outposts*) or connect them via edges (e.g. build *bridges*). Note also that more than one traverser can execute their traversals (with the same or different expressions) at the same time on the graph asynchronously – which is both allowed by the graph computing model as well as currently available implementations of graph engines (see Section 5.2.1). For example, a traversal can be written which, executed as one sequential or many asynchronous traversers, mutates the structure of a graph from an initial single-dimensional line to a two-dimensional small-world graph (see Figure 5.6). Depending on the graph domain, the same can be generalized to any number of dimensions.

```
g.V().aggregate('x').as('a').
  select('x').by(unfold().sample(1).by(both().count())).as('b').
  where('a',neq('b')).addOutE('a','.','b')
```

(A) Small world graph traversal in Gremlin, which results in tranformations of graph structure as shown below.

|V|	Diameter	Average Path	Clustering Coefficient
99	99	34	0
100	55	19.7168316832	0.024
150	9	3.1775717439	0.125

(B) Graph geodesic measures of graphs depicted below.



(C) Graph transformations resulting from running traversal of Figure 5.6a on graphs from left to right. Adapted from Rodriguez (2015a).

FIGURE 5.6: Folding a line into small world graph via graph traversal.

In this way, graph computing can support the creation and navigation of metastability landscapes (Section 3.2.3 and Figure 3.9), stigmergic computing (Section 4.5) and processes of progressive determination (Section 3.2.4), which are instrumental for implementing the open-ended decentralized computing model and the scheme of synthetic cognitive development.

5.2.4 Navigating infinite data structures

Recall that open computing (Section 4.2.5) is described in terms of the following properties:

i. readability of inputs and outputs is determined during the computational process and not *a priori*;

ii. the computation graph, at least partially, is determined by self-organization happening during computation, rather than being explicitly settable by a user *a priori*;

iii. it may be unique and non-repeatable;

iv. it should be possible to construct such a system physically;

v. the computation process may never halt (i.e. can be unbounded in time) yet it should be possible to extract partial or incomplete results at any chosen moment.

Let us see how graph computing allows these requirements to be implemented.

Readability requires that the input of computation has to be determinable and representable in a finite and reasonably compact data structure which can be passed into a computer program. This is problematic due to the sheer volume and velocity of data with which information technologies are currently dealing. Open computing conceives processes that can take as input data structures which are not completely defined at the start and it is therefore able to *operate on the unknown*. Likewise, vertex-centric spreading activation on graphs allows programs to operate on undetermined, incomplete, infinite and changing data structures[21].

Graph computing is *flexible* and non-repeatable since graph traversals may return different results when executed at different times. Note again the difference between graph *traversal*, which is the logic of expression, and *traverser*, which is the actual process that executes it. First, traversers that start from different vertices will obviously walk different paths. Second, since a graph may change during the execution of a traversal (due to mutations caused by other traversals), traversers that are initiated at different moments may take different legal paths in the graph, even if started from the same locality. Third, in graph computing, the computation graph (i.e. the legal path taken in the graph) is a result of the "join" between the description of graph traversal and graph structure that is not defined before the operation of "joining". Fourth, it allows a time limit to be set for a traversal, this way effectively "cheating" the *halting problem* (see discussion on page 92) and returning different results as a by-product. The two latter points deserve a more in-depth discussion.

[21] Issues caused by huge volumes and velocities of data are also dealt by the emerging field of *stream computing* (Assunção, Veith, and Buyya, 2018; Joseph, E.A., and Chandran, 2015) which is based on processing flows of data rather than complete and static data structures.

The mechanics of a graph traverser, which gives rise to semantically constrained spreading activation, is explained by the concept of *grammar-based random walker in semantic network* developed by Rodriguez (2008a) which considers representing both graph and traversal in terms of grammar. A grammar Ψ is equivalent to an expression of a graph traversal (see Figure 5.4), which can also be represented as a graph (see Figure 5.5). A semantic network is a graph G_n. Ψ and G_n are defined with respect to their ontologies, which are of course related – since grammar ontology has to be able to define a legal path in a graph that respects network ontology (see Figure 5.7).

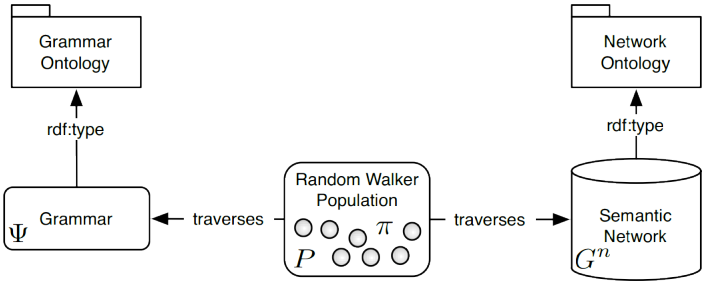


FIGURE 5.7: The grammar based random walker architecture. Adapted from Rodriguez (2008a).

5.2.5 Interaction of subjective perspectives

The result of a particular traverser T is a sub-graph G^{ψ} which is a special kind of "join" of G_n and Ψ (see Figure 5.8).

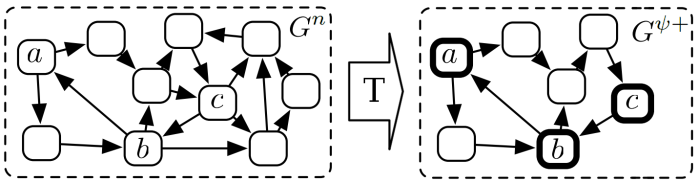


FIGURE 5.8: G^{ψ} as the Ψ-correct subset ("join") of G_n. Adapted from Rodriguez (2008a) with changes.

An important take-away is that G^{ψ} cannot be achieved by G_n nor Ψ alone, but only by their interaction. Consider a conceptual interpretation of such interaction in the context of Chapter 2. In this sense, Ψ represents a "subjective view-point" of an intelligent agent (traverser T) which is making sense of its sensory space G_n. The result of this computation is an agent's subjective perspective of an environment. Different agents in a population ($T \in \pi$) can have different view-points (traversal expressions as well as starting locations) resulting in different models G_T^{ψ} of the sensory space, environment or reality. This interpretation shows how graph computing can be leveraged to account for the aspects of model building discussed in Section 2.2.3 and ecological rationality.

Recall that ecological rationality (Section 2.3.2) is a match between mind and environment rather than an ideal human reasoning and probabilistic inference. The realization of the ecological rationality principle in artificial intelligence is proposed by Wang (2005) in terms of the *Non-Axiomatic Reasoning System (NARS)*. NARS is based on experience-grounded semantics, in which truth and meaning are defined not as objective notions, but according to a system's experience. Experience can substantially differ among systems, even if they operate in the same framework, and therefore interaction between subsystems in the context of a decentralized framework is the only valid way to align subjective meanings. In this way, interaction among subsystems gives rise to system-level meanings. Besides the concept of ecological

rationality, NARS is associated with many concepts related to open-ended intelligence and open-ended decentralized computing, including enaction (Section 2.3.7), sense-making (Section 2.3.8), interactive computation (Section 4.2.5) and more.

Rodriguez and Geldart (2008) and Rodriguez and Neubauer (2011) have developed evidential path logic and path algebra for implementing non-axiomatic evidential logic with grammar walk algorithms on multi-relational graphs. The evidential path logic enables reasoning using arbitrary, partial and contradictory knowledge while supporting a tractable approximate reasoning process. Furthermore, these algorithms can be executed considering available resources and only on those areas of the graph (i.e. sub-graphs) where they are deemed necessary. This way, graph computing allows stigmergic computing to be implemented by concurrently executing multiple inferences in the same framework where graph traversers with possibly different traversing logic interact directly or indirectly while walking the same graph.

5.2.6 Implicit auto-approximation

The interaction ("join") between G_n and Ψ, while being well defined mathematically, computationally is a timebound process, having a certain time that it takes for a traverser T to walk a graph. Since the halting condition of a traverser is defined by the interaction of graph G_n and traversal grammar Ψ, while the structure of the graph is subject to mutations that can be non-deterministic, there is no strict general guarantee that a traverser has a finite execution time. Therefore, in order to deal with the "halting problem" (see page 92) graph computing allows a cut-off threshold to be set. If the time of execution reaches this threshold, the traverser is stopped and its current state is returned as its result, even if it does not comply with the halting criteria defined by the grammar Ψ. A state of traversal is the set of vertices and edges at which the traverser is located at the moment, plus all accumulated data structures if defined by the grammar. While this may appear as a trick in order to cheat the "halting problem", actually it is an important property of graph computing. In optimization problems, this property provides the capability of *auto-approximation*, where criteria of approximation can be defined during execution time, depending on the context of the computation.

Auto-approximation is the computational method for finding out approximate solutions to problems (Shang and Yu, 2014) without knowing or referring to exact solutions in the first place. Very often – especially in machine learning, artificial intelligence, embodied robotics and "big data" analytics – the cost of exact solutions to problems is neither practical nor feasible. When these problems are represented in terms of graph computing, reaching an exact solution may mean executing long traversals where traversers may walk hundreds of millions of vertices and billions of edges. Approximate solutions, which often are good enough for a task at hand, may require orders of magnitude less steps to complete. For example, the PageRank (Page et al., 1999), which is a popular graph centrality measure, is defined recursively, where a vertex is central if it is connected to other central vertices. In order to calculate the exact PageRank measure of each vertex in a graph, all vertices should be traversed, which, in the case of very large graphs (e.g. internet) is prohibitively computationally expensive. Yet, most often, what is needed from graph centrality measures is the *ranking* of vertices by their importance in the network rather than

exact measure. Many fewer iterations may be needed to achieve "good enough" measures for that purpose (Rodriguez, 2015b).

Auto-approximation concretely realizes the principle of ecological rationality internally to computation. Since what is rational can be determined only by considering the environmental context and situation, it may make more sense to achieve "good-enough" approximate results in less time than exact results in a much longer time – and that could be a matter of survival and further development for an intelligent agent. I consider this property consequential for applications related to artificial general intelligence.

5.2.7 Decentralized indexing

Indexing is an important technical aspect of graph computing and the operation of decentralized networks in general. Current practical organization of the web (semantic or not) is most instructive in this sense, considering especially that it is an evolving rather than *a priori* designed system. Most people using the web today access it through *search engines*, which are on-line services allowing users to search content that is scattered through the network (or graph) of servers and webpages using keywords, natural language expressions or structured queries. Since the web is decentralized and open, nodes and links can be created, deleted or have their content modified in an asynchronous manner. Search engines allow navigation through this dynamic "giant global graph" by employing two main components – the web crawler and an index:

- *Web crawler* (also called *bot* or *robot*) is a special program which autonomously browses the web with the help of hyper-links and sends the information about each visited page to the search engine. In graph computing terms a web crawler is equivalent to a graph traverser walking the graph with a specific graph traversal expression.
- *Index* is a central database where all information from web crawlers is collected, indexed and stored for querying. All search engines then have a user interface – usually a webpage – through which users can query the database by content of interest and then follow provided hyper-links. Modern search engines also have APIs, allowing their database to be queried by other nodes in the graph without human intervention[22].

Qualitatively speaking, search engines and their indexes are also nodes in the web graph, alongside other servers, programs and devices. Yet they have two notable quantitative differences: (1) a much larger degree of connectivity to other nodes (which is the whole point of search engines – i.e. providing a gateway to other nodes) and (2) more computing resources (storage space and processing power). These are the *supernodes*. In network science and graph theory, a supernode is a vertex of a graph that has a disproportionally large number of connections to other vertices (Rodriguez, 2012). The existence of supernodes in real world graphs is related to the empirical phenomenon having many names – Paretto principle, power-law distribution, scale-free network – carrying the same meaning: that the influence in most real complex networks is not equally distributed among all participants. Moreover, the number of nodes with certain influence is inversely related to their influence values.

[22] e.g. Google Custom Search API, Bing web Search API or Twitter Search API.

In other words, most of the nodes in the complex network are unimportant and only a few are very influential (Barabasi and Frangos, 2002; Kshemkalyani and Singhal, 2008).

Relating notions of index in graph computing and supernode in network science allows distinctions to be formulated between centralized (or global), distributed and decentralized indexes, as pictured in Figure 5.9a:

i. *Centralized*, *global* or *graph-centric* index contains information about all vertices or edges in the graph, and therefore gives rise to a centralized network structure by providing a routing path for all nodes to all other nodes. Note, however, that global indices can index only a certain chosen aspect of a complex graph structure (e.g. a certain property type of all vertices or edges), and therefore are not "omniscient" about the graph, but represent a single global perspective of a chosen aspect.

ii. An index which operates as a global one yet is located in several tightly cooperating nodes is *distributed*, giving rise to a distributed network structure. All these nodes represent the same global perspective of a graph. A notable example of such an index is the *distributed hash table* (DHT) technology, used for global indexing of peer-to-peer networks without changing their architectural structure.

iii. *Decentralized*, *local* or *vertex-centric* indices are the ones that index only the local properties (e.g. edges) of a node. Obviously, this type of indexing gives rise to decentralized networks and is used in random walk searches (Kshemkalyani and Singhal, 2008; Sandberg, 2005).

(A) Centralized, distributed and decentralized indices as network structures, (see also Figure 4.4c on page 99).

(B) Graph-centric and vertex-centric indices (Rodriguez, 2016b).

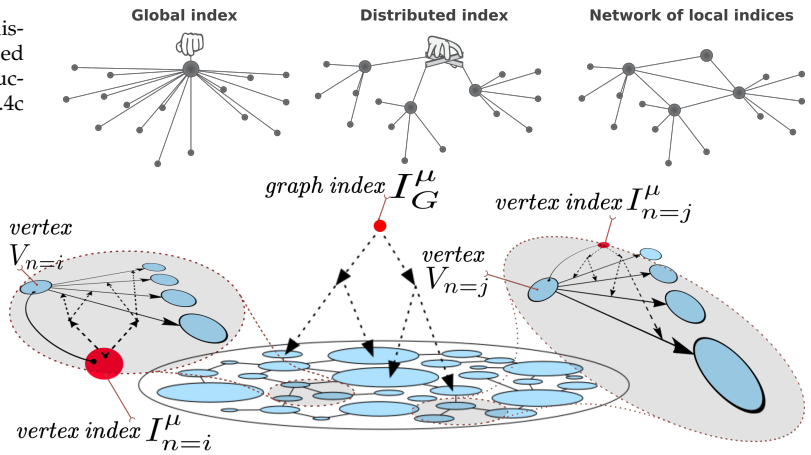


FIGURE 5.9: Graph-centric and vertex-centric graph indexing gives rise to centralized, distributed and decentralized network structures.

From the standpoint of graph computing, vertex-centric indexes allow a link for further traversal to be chosen very quickly. From the standpoint of network science, supernodes make graphs more connected – allowing each node to reach each other via shorter paths and less time. Scale-free graphs of huge size are known to be very well connected – e.g. the diameter of the world wide web subgraph with 800 million nodes is only 19, whereas human social networks of billions of nodes are believed to have a diameter of about 6 (Albert, Jeong, and Barabasi, 2000). Recall that the

diameter of a graph is the longest path without cycles that can be found in it[23]. In the context of internet and the web, better connectivity intuitively, while somewhat simplistically, means better accessibility to information for every participant. Indexing therefore can be seen as a way to introduce specialized supernodes which effectively reduce the diameter of a graph and make information propagation between nodes more efficient. Another aspect of indices in general and search engines in particular is that they provide a mapping between node content and address for routing purposes. Actually, modern search engines introduce a kind of content addressing into the web – often one does not even notice the actual URL of a server or webpage one follows from a search engine's page.

In short, supernodes allow for more efficient navigation in the graph (Hadaller, Regan, and Russell, 2005). Furthermore, as Albert, Jeong, and Barabasi (2000) show, the existence of supernodes in the graph makes it more resistant to random attacks, which arguably is one of the reasons why internet is so robust despite the unreliability of each individual link. On the other hand, the existence of supernodes also has drawbacks: they make a network more vulnerable to targeted attacks (ibid.). Also, in graph computing, supernodes may lead to performance problems due to large number of traversers checking all connections and clogging the system's resources. Finally, from the practical perspective of governing decentralized systems, such as internet, the web or economic networks, supernodes may introduce a self-reinforcing inequality and confirmation biases and reduce the diversity needed for healthy evolution of a system. "The supernode problem" (Rodriguez, 2012) therefore provides a network-scientific and graph-theoretic perspective to the more general and conceptual problem of self-organizing complex adaptive systems. The relative proportion of supernodes in such systems may crucially affect their performance in different environmental situations[24].

On a conceptual level, there are no clear-cut boundaries between centralized, decentralized and distributed indexes, as they represent a continuum between centralization and decentralization, as discussed in Section 4.3. Modern computing systems provide implementations of all types of indexes[25]. Open-ended decentralized computing, however, requires a *fluid* indexing scheme, where global indices and centralized network structures construct and dissolve themselves organically and constantly as a result of bottom-up self-organization. Therefore, open computing emphasizes the process of how indices and network structures morph to each other rather than their static forms.

The dynamics of the web illustrates such self-organization: all search indexes through the history of internet[26] have emerged bottom-up and grew to the dominance and supernode status without an initial preferential position. This exemplifies how every node in a network potentially can become a supernode, provided it has computational resources to traverse the graph and store collected information. Likewise, nodes in a graph computing context can autonomously grow centralized indices by issuing graph traversals which walk the whole graph resulting in global

[23] More precisely, graph diameter is the longest shortest path in a graph, or the distance between the two furthest away nodes (Barabasi, 2013).

[24] "A relative proportion of supernodes and their importance" in a network can be measured by the power-law exponent (Easley and Kleinberg, 2010)

[25] E.g. actor framework akka.io, as much decentralized as it is, provides a hierarchical addressing of all actors in an actor system, which is a kind of centralized index. In graph databases world, DataStax Enterprise Graph supports both graph-centric and vertex-centric indexing.

[26] W3Catalog, Infoseek, Yahoo, Altavista, Google, etc.

self-organized structure, which would influence overall dynamics of a graph.

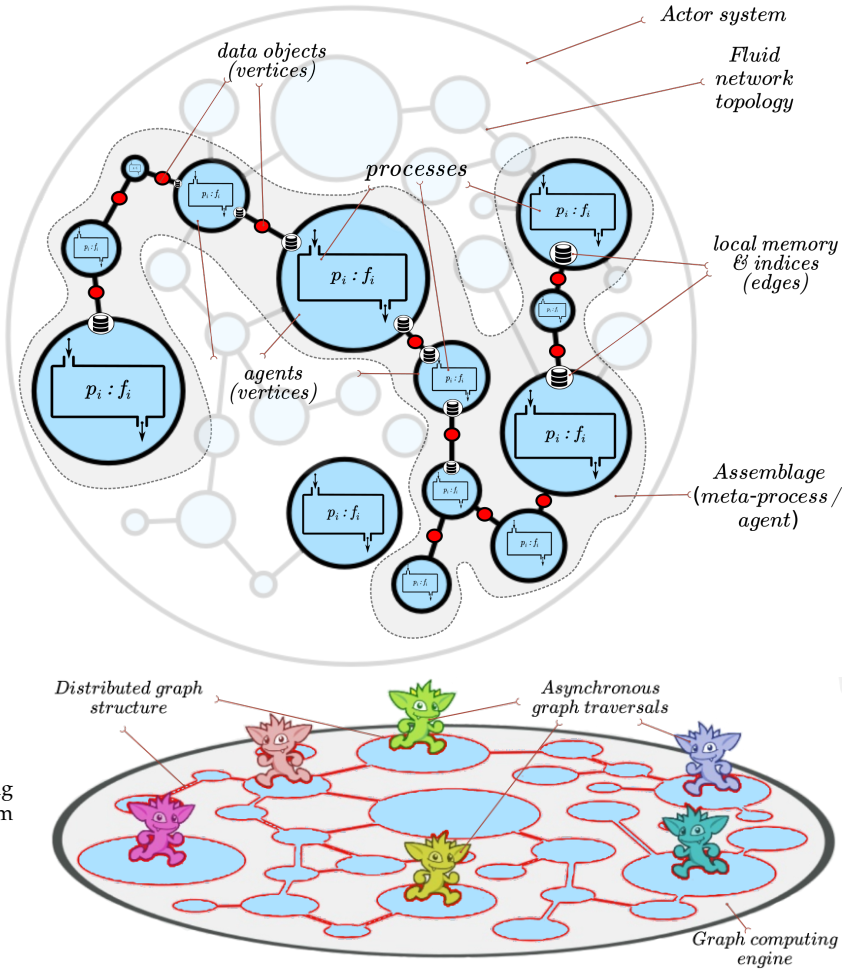
5.3 Architecture for open-ended computing

The architecture for open-ended decentralized computing unites the actor model (Section 5.1) and graph computing (Section 5.2) into one framework. The actor model enables massively scalable implementation of a population of independent and heterogeneous computational processes, and their custom behaviours. Actors can have wildly different computational complexities and resource requirements. Computational processes of actors communicate with each other by passing messages which encapsulate commands and data structures. The totality of all traces of messages constitute a dynamic message-passing graph – a shared data structure – where every actor is also a vertex. Graph computing enables actors to access and change this graph by executing vertex-centric graph traversals based on their unique perspectives and custom behaviours. The message-passing graph, while best reasoned about in terms of a logical structure, is actually scattered across local memories of actors and does not constitute a single object. Therefore, it is not accessible directly, but only via the fact that an actor which accesses it is itself a part of the graph and can issue traversals from its location and via its links with immediate neighbours. Most importantly, the notion of message and graph traversal are different aspects of the same communication event. This fact alone unites the actor model and graph computing into one framework by allowing actors to specify legal paths of messages. In this way messages (i.e. information and control flow) propagate not randomly or directly, but depending on internal logic and the local graph structures, which develop and change because of other messages. Recall that the actor model allows the operation of the framework to be radically decentralized and asynchronous, while graph computing allows for local computations. The framework enables the bottom-up emergence of computation graphs to be realized as persistent assemblages (sub-graphs) of computational processes and, in general, stigmergic computing and progressive determination. This is a high-level description of the architecture of open-ended decentralized computing in a nutshell. Figure 5.10 depicts the architecture by showing the actor system (5.10a) and graph computing engine (5.10b) as integrated aspects. Each named component of the architecture is described further.

All computational components are implemented both as actors and as graph vertices which allows to leverage both flexibility and asynchrony of the actor model of computation and indirect communication via evolving shared data structure enabled by graph computing. There are two generic types of component in the architecture, which can be further specified based on the domain model requirements: *data* and *processes*. These components are specified by the architecture because they are fundamental elements of computing[27]:

- *Agents* are actors which encapsulate elementary computational processes. They are also vertices. Every process p_i implements a function f_i which takes data object o_x as input and produces data object o_y as output (see Section 4.5.1).

[27] Note that graph computing allows both graph traversals (which are processes) and data to be represented within the same graph structure (Rodriguez, 2016a), in this way blurring boundaries between data and process. The architecture of open-ended decentralized computing, as formulated currently, does not benefit from this option, but it is an interesting possibility for the future.



(A) Actor system.

(B) Graph computing engine. Adapted from Rodriguez (2016b).

FIGURE 5.10: Open-ended decentralized computing architecture.

Note that an elementary computational process is necessarily sequential, indivisible and represents the lower-most scale of the scalable structure of a computation graph (see Figure 4.6). As actors, agents can send and receive messages from other agents and react to them. As vertices, they issue graph traversals for exploring their neighbourhood structures.

- *Data objects* are actors which encapsulate data and are also vertices of a graph. While agents can implement any computational process, data objects implement a very specialized function, which can output stored data or some description of it if "asked"[28]. Data objects are therefore special cases of agents, yet still useful to define for didactic purposes.

Agents and data objects as vertices of the graph become linked to each other and form a *fluid network structure*. This structure is neither persistent nor ephemeral, in strict computer-scientific terms, but exists somewhere in between. The network structure emerges from traces of message passing among actors, but in a selective manner: not all messages leave traces and some traces can persist longer than others. It is an important feature of open-ended decentralized computing that the network

[28] A data object can also be considered a process which implements a function with two possible inputs and two possible outputs: e.g. if it gets input 1, it outputs data stored in its memory. If it gets input 0 it outputs the description of the data that it holds.

structure is represented in a distributed manner and is an emergent phenomenon. The emergence of the network structure is facilitated by every actor-vertex having a *local memory* which holds an index of links to other actor-vertices of the system (agents or data objects) and properties of these links. First, this means that there is no notion of global memory, where the network structure and information about links between actor-vertices are stored. This of course means that there is no explicit pointer to the structure and it can be accessed only via one of the actor-vertices. Second, the fluidity of the network structure is determined by local interactions among actor-vertices and is a result of certain "agreed-upon" or "negotiated" patterns of interaction – i.e. message passing. Third, the support for graph computing via vertex-centric graph traversals allows actors-vertices to explore the distributed network structure beyond their immediate neighbourhoods. A traverser in the open-ended decentralized computing system is a message which encapsulates the traversal expression and path algebra logic. When an actor receives such a message, it executes the current traversal step and passes the expression to actors listed in its local memory which satisfy the conditions of the step. This way the message-traversal walks a legal path in the distributed graph structure and realizes semantically constrained spreading activation in infinite data structures (see Section 5.2.4). Note that, technically, the absence of a globally defined and explicit network structure means that graph computing does not need to be implemented via graph databases or to use them in any manner. This is an important aspect of the open-ended decentralized computing model that allows it to embrace **radical decentralization** as its primary architectural principle.

The mechanics of vertex-centric graph traversal over a decentralized graph structure is explicated by Figure 5.11. Figure 5.11a repeats the running example of graph traversal expression in Gremlin while Figure 5.11b shows the mechanics of this traversal over the decentralized graph structure composed of small data structures localized at each vertex.

The decentralized traversal works as follows:

0. the traverser starts at the counter value of zero at the condition indicated by *step 0*, which is vertex with *id:7*; the counter value is incremented by one: $c = c + 1 \rightarrow c = 1$;
1. next, it checks if local index at vertex *id:7* contains outgoing edges with label *dirtRoad*, which is the condition of *step 1*; since three such edges exist, the traverser branches and walks to vertices *id:15*, *id:32* and *id:56*; the counter value is incremented: $c = c + 1 \rightarrow c = 2$;
2. next, the three traversers at reached vertices *id:15*, *id:32* and *id:56* check if their names are legal according to the condition of *step 2* (i.e. that *name:outpost*); since all three vertices are in legal path, counters are incremented for all three traversers: $c = c + 1 \rightarrow c = 3$;
3. next, the traverser checks if local indices contain outgoing edges with label *bridge* which is the condition of *step 3*; vertices *id:15* and *id:56* do not contain such a further legal step, and therefore traversers die at these vertices; the only remaining traverser continues from vertex *id:32* by branching again to vertices *id:65*, *id:66* and *id:67* along edges with name *bridge*; counters are incremented: $c = c + 1 \rightarrow c = 4$;
4. next, traversers at reached vertices *id:65*, *id:66* and *id:67* check if their names are legal according to condition of *step 4* (i.e. that *name:town*); since all three

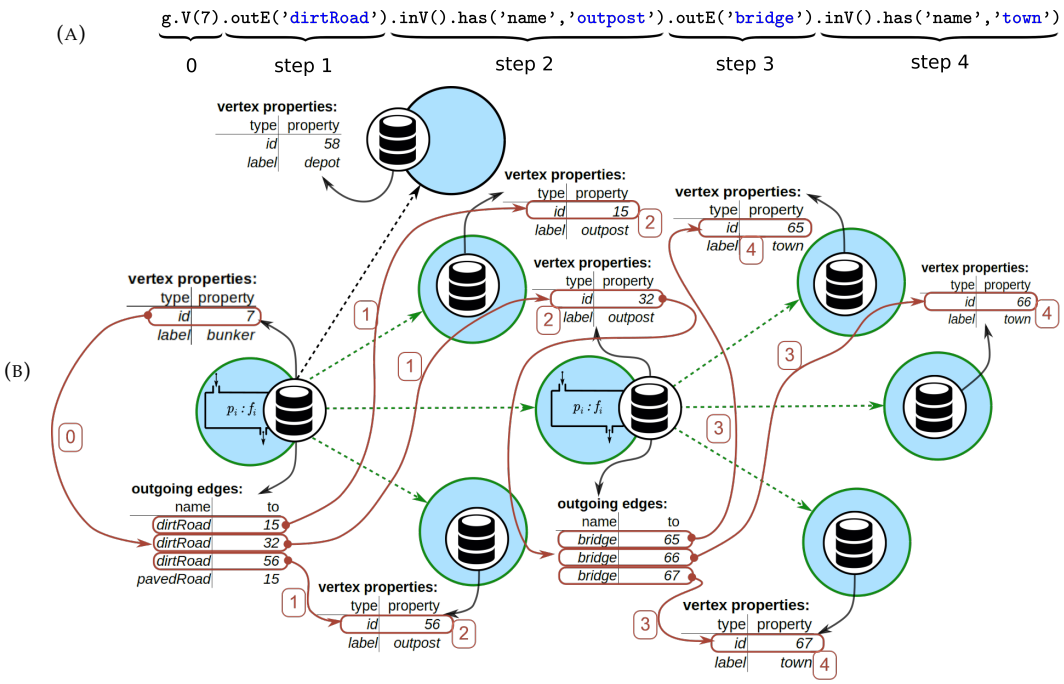


FIGURE 5.11: Decentralized graph traversal (5.11a) and structure (5.11b). Vertex-centric graph traversal is implemented with grammar-based traversers checking next legal steps in their walk against local data structures. Traversers are then sent as messages through legal steps, their counters incremented by one and the procedure is repeated at arrival vertices until there are no more legal paths in the graph or the halting condition is met. Edges and vertices selected by the traverser as legal local steps are marked with the green.

vertices are on the legal path, and *step 4* represents the halting condition of the traversal, the traverser is stopped and these vertices are returned as the result.

Note that this result is equivalent to the graph traversal illustrated by Figure 5.5. The graph structure is created and sustained in a distributed fashion by each vertex holding an index of its outgoing edges and their properties. This allows all graph computing semantics and capabilities to be preserved in a completely decentralized system based on interacting independent actors, including mutating the graph in terms of creating new links and vertices, and changing their properties. Further recall that all vertices are also actors and as such can implement any elementary computational process complete in the Turing (1937) sense which can be triggered by the traverser. A traverser, besides traversal semantics, can carry custom data structures to be processed (read and written with changes) at any vertex – which allows the implementation of graph algorithms that process information collected from traversed vertices. Finally, the actor model augmented with the decentralized graph computing allows for the emergence of assemblages of elementary processes as higher order computational graphs and in this way realizes the self-organization of unorganized machines in the Turing (1948) sense (see Figure 4.5).

Last, but not least, open-ended decentralized computing architecture allows the actors' mobility, location awareness and therefore computational economy. It is perfectly conceivable (yet not trivial) that a mechanism could be implemented where actors (data objects or processes alike) can "transport" themselves to a different location of the graph where local costs of connectivity are more favourable for the

computational process of the computational graph of the assemblage that they represent. It is also perfectly conceivable that graph traversals could be implemented which are aware of such cost constraints.

This concludes the discussion of the open-ended decentralized computing model and architecture as a combination of the actor model of computation and graph computing. Some high-level technical aspects of the architecture are discussed further, before proceeding to the description of example implementation – the *offer networks* – in Section 6.

5.3.1 Implementation guidelines

The nature of open-ended decentralized computing dictates that computing processes cannot be directly observed or controlled from a single point in software architecture. This contrasts with the usual practices of software development. While radical decentralization is the main principle of the open-ended decentralized computing concept and architecture, observability and influence are still required for software built on top of this computational model to be of any practical value. Therefore, in order to provide observability over computational processes, the software framework based on the open-ended decentralized computing model features two fairly separate subsystems – simulation engine and analysis engine.

- *Simulation engine* hosts the implementation of all aspects of open-ended decentralized computing: (1) heterogeneous agent's logic, (2) actor framework, (3) graph computing engine and persistent network topology (distributed or otherwise).
- *Monitoring and analysis engine* collects information of events happening in the network and performs real-time or post-mortem analysis or visualizations as required by software design requirements.

The two subsystems communicate via events that are issued by components of the simulation engine (actually, every actor). The granularity and type of events are determined by each domain model and the requirements of software design. The monitoring and analysis engine is a separate subsystem, most often physically located on separate hardware which catches all events for storage and analysis (see Figure 5.12).

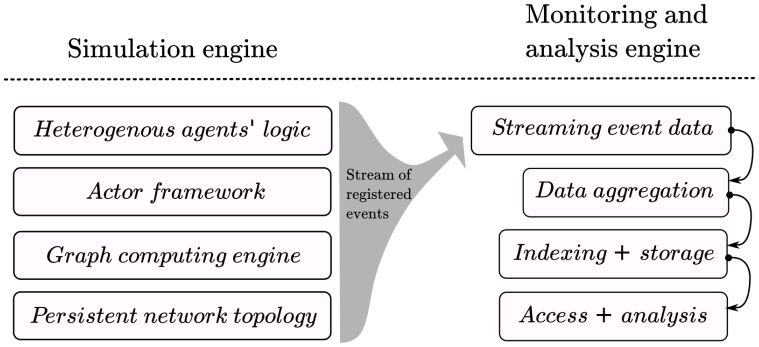


FIGURE 5.12: Simulation and analysis engines of open-ended decentralized computing architecture.

Note that very often the sheer number and velocity of events issued by the simulation engine, which may host millions of asynchronously executing actors, require the monitoring and analysis engine to be of at least the same computational capacity as the simulation engine itself.

5.4 Summary of the chapter

This chapter proposes a software architecture and further defines low level semantics for realizing the computational model of open-ended decentralized computing based on the notion of stigmergic computation. The key aspect of the architecture is the integration of two models of computation: Actor framework and graph computing. With the actor model, this integration achieves four tenets of open-ended decentralized computing: population of independent processes with custom and heterogeneous behaviours and ability of direct interaction between processes via message passing. With graph computing we achieve indirect interaction via a shared medium, fluid evolving structure of the shared medium and, finally, radical decentralization via the decentralized graph traversals.

We first discuss the actor model in terms of its informal as well formal description. This is followed by the formulation of fundamental principle of decentralized computing, involving the choice between computation and communication. We then show the locality and mobility extensions of the model, which enable the conception and implementation of *computational economy* of computational agents. Second, we discuss the emerging field of graph computing by presenting its theoretical and practical aspects in terms of graph databases, graph traversals and vertex-centric spreading activation. Then, the relative properties of graph computing are presented – the ability to navigate infinite and massively scalable data structures, interaction of subjective perspectives and computational processes, auto-approximation and decentralized indexing. Third and finally, we demonstrate how the two computing models integrate into one framework and provide the basis for the emergence of computation graphs as assemblages of agents via the computational implementation of the mechanism of progressive determination.

Note that the architecture of open-ended decentralized computing provides a basis for the implementation of synthetic cognitive development processes, but does not in any way constrain or guide the types of process and dynamics that can be achieved by it, apart from realistic representations of a complex dynamic system composed of history-dependent interactions of population of lower level agents. While the architecture allows the modelling of large-scale decentralized computation, hybrid systems, social socio-economic interactions and many more, the specific dynamics and properties of these systems will have to be implemented above the architecture of open-ended decentralized computing and based on the chosen domain model logics. The next Chapter 6 discusses one possible example of implementation of a specific domain model – a decentralized non-monetary exchange.

Chapter 6

Offer networks: a model of decentralized exchange

Offer networks is a work-in-progress concept of an alternative economy where heterogeneous and independent agents (humans, AIs and more or less simple programs and intelligences) find, negotiate and execute locally and globally beneficial series of hybrid exchanges of tangible and intangible goods. The initial concept was proposed by Goertzel (2015a) in the context of a post-money economy and further explored by Heylighen (2017) in the broader context of decentralized coordination, distributed intelligence and global brain research. Heylighen (ibid.) also proposed an alternative name for such general system: the *synergy web*. Our purpose of referring to offer networks in this thesis is mainly motivated by using it as a domain model for trying out the open-ended decentralized computing architecture described in Chapter 5 in terms of an actual working code and integration of various existing software frameworks. Nevertheless, we first introduce the economic context of offer networks, which is important to grasp for appreciating a case of a complex adaptive and necessarily decentralized self-organizing system.

6.1 Economic context

Established economic theories, classical and neo-classical alike, largely understate or ignore subjectivities and complexities inherent in how individual humans determine the value of goods and services and achieve their exchanges. Social structures are seldom discussed behind the notion of the market as an ideal resource-allocating mechanism in which dynamics is abstracted from the individual behaviours and preferences of its participants (Gode and Sunder, 1993; Jackson, 2006). The ideal market image is perfectly competitive, where all sellers and buyers are homogeneous or, at best, categorized into large homogeneous groups. Such a market (or exchange) is utilitarian and driven by utility maximization principles, which are objectively definable for each group. A canonical example is the financial market, which is carefully stripped of any social interaction by strict regulations. Yet it is a special case which does not apply to economic exchanges in general, nor is it a desirable model of them.

Economic sociology, a term coined in late 19th century, is "the application of the frames of reference, variables, and explanatory models of sociology to that complex

of activities which is concerned with the production, distribution, exchange, and consumption of scarce goods and services" (Smelser and Swedberg, 2005). A central concept in contemporary economic sociology is the embeddedness of economic actions in concrete and ongoing systems of social relations (Granovetter, 1985) – i.e. social networks and sub-networks. Exchange, as the mechanism of coordination of actors and owners of resources, is the outcome of their local interactions via dynamic social networks which evolve in time as a product of the totality of these interactions (Johanson and Mattsson, 1994). Note that this is clearly a case of progressive determination (see Section 3.2.4). Obviously, network science plays an important role in the research methodology of economic sociology. Markets are highly connected complex adaptive systems and understanding them requires integration of ideas for reasoning about network structures, strategic behaviours, feedback effects, reflexivity, and more. From the perspective of open-ended distributed computing, markets are seen as assemblages of economic agents and as platforms for the emergence of further sub-assemblages and local social networks. In economic sociology, markets and strategic interactions in networks can be explored by combining game theory, which focuses on interactions among independent agents, and graph theory, which focuses on their social relations (Easley and Kleinberg, 2010).

Another important assumption of "mainstream" economic theories is the rationality of market actors, including the completeness and transitivity of their preferences. In simple terms, completeness means that market actors always have clear preferences with respect to the choices that they are faced with. Transitivity then basically enables all preferences to be ordered with respect to each other. Despite plenty of evidence against the theory of rational choice, assumptions of rationality seem to be holding their ground (Mandler, 2005) at least in part because it is difficult to build a clearly interpretable model of economy and society without them.

Practically, however, people often do not know their preferences *a priori*, and these preferences are not comparable to each other or variate over time and depending on the context. Consider, for example, a flea market where sellers offer unique items that can have aesthetic, emotional and practical value. More often than not, buyers come to the market with only a vague, if any, idea of an item they are interested in. They mostly make a decision only after seeing a number of items, interacting and negotiating with sellers. Such negotiations are not only technical exercises of matching supply and demand in order to come up with the fair price (as in perfectly competitive marketplaces), but also a social interaction during which preferences themselves individuate. The agreed price of an item may even not be the main aspect of a transaction, and surely may be different depending on individual preferences of buyer, seller or simply a circumstantial result of their interaction. Such dynamics cannot be captured by traditional models of faceless buyers and sellers asking and bidding on quantities and prices of standardized goods. Summarizing, the value of an item of exchange is at least in part subjective, unique and dependent on singular expressions of immediate social interaction, not fully reflected in its monetary value. The idea of offer networks is to reclaim the subjective value of economic exchange, neglected by utilitarian efficiency-directed theories and standardised markets. In the context of large-scale digitalized markets, where the face-to-face conversations of the flea market are no longer an option, there is a need for a computational solution.

In a nutshell, the concrete computational problem addressed by experiments described here is *search and matching*. It is a well established and researched problem in labour economics (Dao, 2011), an aspect of which is matching unemployed persons'

skills with free jobs in economy. The problem of matching job openings with potential employees obviously involves more variables than the salary level. Likewise, offer networks is a call to conceive a market where exchanges are driven by matching the subjective, diverse and multidimensional values of each participant engaged in an exchange rather than reducing them to one all-permeating dimension mediated via single currency (Goertzel, 2015a). Open-ended decentralized computing allows the practical implementation of search and matching in the light of a *radical decentralization* perspective as a process of bottom-up local self-organization, not unlike how real world markets operate. Furthermore, it exposes numerous aspects of fundamental distinction between *centralized* and *decentralized* approaches to the governance of complex, fluid and self-organizing systems – including economics, society, IT systems and artificial intelligence.

With the help of OfferNets we continue to weave the thread which intertwines the computational, conceptual and philosophical aspects of this work. The offer networks model provides a concrete case for a computer simulation based on the computational model that represents a much broader class of complex self-organizing cognitive systems. Note that conceptual, computational and software models of offer networks are strongly intertwined and the boundary between them is fuzzy. For the sake of the discussion's clarity we will refer to the conceptual aspect as **offer networks** and computational / software architecture aspect as **OfferNets**, while interchangeably using both.

6.2 Software architecture

OfferNets is a simulation modelling framework of radically decentralized economic interaction, powered by a diverse network of independently operating and interacting agents. It combines two research and development paths which are tightly related, yet embrace different levels of abstraction:

- *Decentralized computing*: a scalable computing model and a software framework supporting asynchronous execution of heterogeneous processes. These processes concurrently use a shared distributed data structure that allows any mixture of emergent and controlled coordination to be modelled (see Chapter 4).
- *Offer networks* economy: a decentralized economy providing an alternative to purely currency-based exchanges. This economy features a complex network of interactions and optimizes reciprocal exchanges of goods and services by finding agents with compatible or complementary preferences and coordinating their interactions. However, we do not attempt, conceptually or computationally, to model the whole economy here, but restrict ourselves to the demonstration of a specific search and matching problem, as introduced earlier.

Recall that, architecturally, software framework is composed of two subsystems: simulation engine and analysis engine. The concrete implementation of both is explained below. See Figure 5.12 for an explanation of interaction between these subsystems.

6.2.1 Simulation engine

OfferNets simulation engine architecture, software framework and relations to the implementation of the offer networks domain model are illustrated by Figure 6.1:

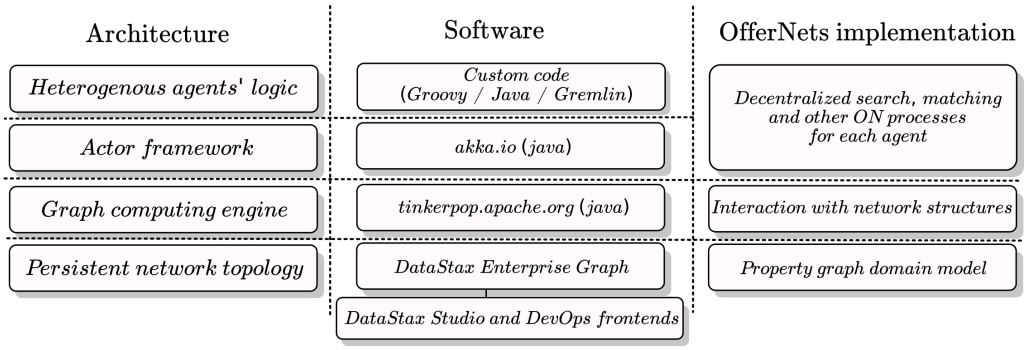


FIGURE 6.1: Relation between architecture, software and the offer network domain model implementation of the simulation engine.

The four layers of architecture are implemented as follows:

i. *Heterogeneous agents' logic* is written in custom code using Groovy[1], Java[2] and Gremlin[3] programming languages. The choice of languages is mostly constrained by the choice of actor framework and graph computing framework which are both JVM based;

ii. The chosen *actor framework* is Akka[4], which is an implementation of the actor model on Java Virtual Machine and is a toolkit for building highly concurrent, distributed and resilient message-driven applications on Java and Scala[5]. Depending on the complexity of actor code, it may support up to 50 million messages per second on a single machine, 2.5 million actors per 1 GB of memory, actor systems spanning multiple machines and a persistence layer.

iii. *Graph computing framework* is the Apache TinkerPop[6], which is also the home for Gremlin graph traversal language. The framework is open source and vendor-agnostic, meaning that it can be integrated with many data sources, database systems and programming languages of the open source and commercial world, including massively scalable cloud-based technologies[7]. Apache TinkerPop is a matured, but actively developed, de-facto graph computing standard with the third major stable version at the time of writing.

iv. *Persistent network topology* (equivalent to the "data source" in TinkerPop's vocabulary) is implemented using DataStax Enterprise (DSE) Graph[8], which is a distributed and scalable across multiple machines graph database. While the DSE Graph is built with and natively supports TinkerPop technology, it is the

[1]https://en.wikipedia.org/wiki/Apache_Groovy
[2]https://en.wikipedia.org/wiki/Java_(programming_language)
[3]https://en.wikipedia.org/wiki/Gremlin_(programming_language)
[4]https://akka.io/
[5]https://www.scala-lang.org/
[6]http://tinkerpop.apache.org/
[7]For the current list of TinkerPop-enabled providers and integration introduction, see http://tinkerpop.apache.org/providers.html
[8]https://www.datastax.com/products/datastax-enterprise-graph

most easily replaceable layer of the stack. Furthermore, from the perspective of the computational model, a graph database is not a necessity, since graph computing can in principle be implemented in a completely decentralized manner without a central database (see Figure 5.11) by using other means for ensuring persistence (e.g. Akka persistence layer). The usage of DSE Graph in the current version of OfferNets is a matter of convenience, since no graph computing engines decoupled from graph databases currently exist.

As illustrated in Figure 6.1, the actor model (via Akka) and agents' logic supports the implementation of OfferNets decentralized search and matching, as well as any offer networks related algorithms which can be implemented independently on each agent. The graph computing engine (via Tinkerpop) enables agents to define a domain model and interact with the network structure, the persistence of which is supported by the DSE Graph. Note that the computational model and software architecture, described in Chapter 4 and Chapter 5, does not constrain the choice of actual software for any layer of the stack, given they support the actor model and graph computing.

Further, the OfferNets domain model is specified as a property graph schema in terms of types of node, their properties, types of edge, their properties and processes defining graph traversal and mutation constraints. Beyond that, every agent operating in the network is allowed to implement any process. Processes which require interaction with the OfferNets graph are implemented as graph traversals; other processes – as regular algorithms using general purpose programming languages.

This decentralized computing model and architecture allows us to implement, test, deploy and observe the evolution of a very large number[9] of computational processes interacting and coordinating directly or indirectly within the same ecosystem. The challenge is then to define concrete processes, design their interaction and fine-tune the system to the preferred dynamics of the offer networks economy.

6.2.2 Monitoring and analysis engine

Simulation modelling requires the collection and analysis of information about events happening in the system during runtime. Since a decentralized computing framework by definition does not contain a single point of access to the system, we have built a specialized engine for collecting and handling large amounts of streaming data coming from many sources (i.e. single actors potentially scattered across multiple machines). The basic principle of the engine is based on issuing monitoring messages on behalf of each agent and then catching and indexing them into a single (but possibly distributed) database. The technical basis of the engine is ElasticStack[10] – an integrated streaming data management and analysis solution (see Figure 6.2).

The monitoring pipeline is fully distributed, with the possibility to be scaled to multiple machines and is tolerant to failures and restarts of each component separately. Likewise, the simulation engine is readily scalable to multiple machines depending on the required load for simulation or production environments. Both provide real time monitoring capabilities across all machines via web front-ends. Additionally, real time network, agent activity monitoring and event capturing is

[9] The scalability of the framework is of course dependent on the amount of dedicated hardware resources.

[10] https://www.elastic.co/

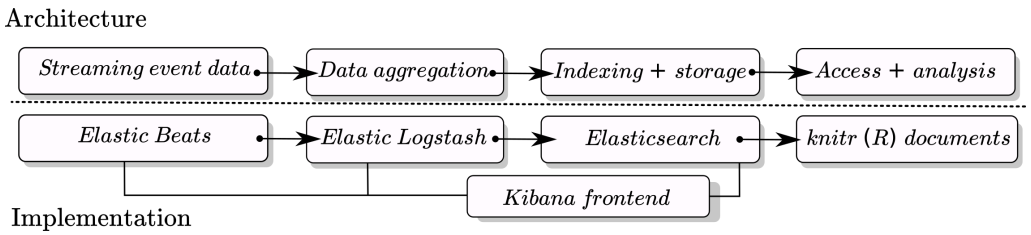


FIGURE 6.2: Architecture, software and implementation of monitoring and analysis engine.

available via custom-based web front-ends accepting data streams from other parts of the infrastructure.

6.2.3 Simulation modelling

Designing the dynamics of OfferNets is a simulation modelling research programme. It amounts to conceiving, implementing and running computational experiments on the simulation engine and then analysing data collected via the monitoring engine. Due to the large parameter space and many simulations needed for exploring it properly, this is a computationally intensive process. Furthermore, it is an open-ended process in the sense that every simulation raises questions and informs the set-up of the next one, thus iteratively perfecting both computational infrastructure and the domain model.

In the following sections we provide details about simulations aimed at comparing centralized and decentralized search algorithms on the same graph structures. Analysis, interpretation, and design for new experiments are compiled into the electronic laboratory notebook[11] using *R markdown*[12] living documents. The raw data is publicly available at the project's GitHub repository[13].

6.3 OfferNets: informal specification

Our goal is to design a software framework for the system pictured in Figure 6.3 and explore stigmergic cooperation between agents by running computational experiments. The system should allow matches of *offers* and *demands* (i.e. data *items*) to be found in an economic exchange network of *agents* in a way that enables each agent to extract maximum value from its exchanges – with respect to an individual measure of value – and by that contribute to the total increase of value created in the system. Already here we can start to appreciate the distinction between the decentralized aspect – the measure of value defined individually for each participant – and the centralized one – some sort of measure of "universal" value in the system[14].

[11] Electronic laboratory notebook is accessible at https://singnet.github.io/offernet/public/elabnotebook/

[12] https://rmarkdown.rstudio.com/

[13] Project's GitHub repository is accessable at https://github.com/singnet/offernet

[14] This distinction also presents a methodological difficulty: if every agent in the system has a different notion of "value", which, moreover, may change in time, then the notion of "universal value" is difficult to define. This means that it is not trivial to measure the overall "progress" of a decentralized system. These are open issues in relation to the *theory of value*, further discussion and development of which is outside the scope of this work.

Figuring out how these aspects relate and influence each other is the holy grail of research in the domain of collective intelligence and distributed cognition (Heylighen, 2011, 2013).

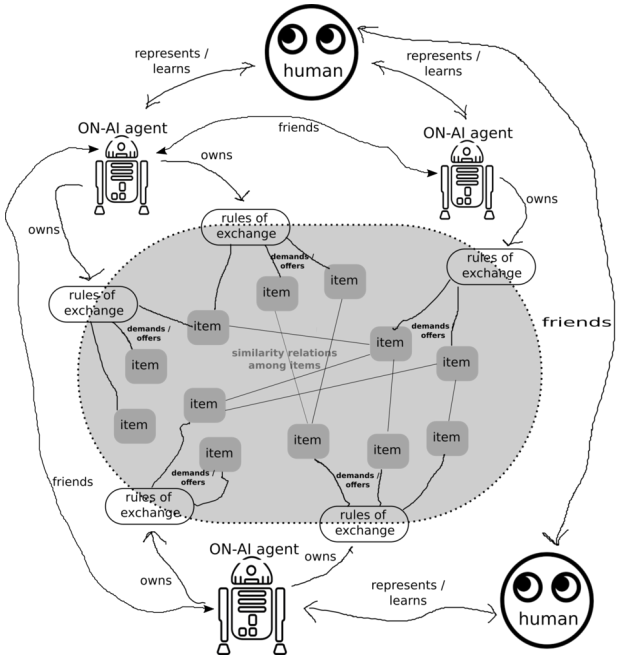


FIGURE 6.3: Conceptual architecture of OfferNets.

In OfferNets, humans can be represented in the network by one or more *ON-AI agent*. These computational agents, with more or less complex behaviours, can learn certain preferences of a represented human. Since there is no central authority in the network, humans can connect to it only via other humans, i.e. via a recommendation. Every human can create as many *ON-AI agents* as wished. The sole goal of an ON-AI agent is to announce preferences for the network and find ways to satisfy them. Preferences are satisfied when an agent finds other agents in the network (representing other humans) with opposite preferences. In OfferNets, preferences are expressed in terms of preferred exchanges of *items* – which in the real world could be goods, services, data or procedural knowledge. To that end, each ON-AI agent can own a set of *works*, which encode preference relations among the set of items and define items that are demanded and items that are offered (see Figure 6.4). The search and match of an ON-AI agent is successful when, given it *offers* an item, a similar enough item *demanded* by another agent on the network is found (this of course also works the other way).

ON-AI agents relate to each other via *knows* links which define the $agent \xrightarrow{\text{knows}} agent$ social sub-graph of OfferNets. Likewise, *items* can link to each other via *similarity* relations that define $item \xleftrightarrow{\text{similarity}} item$ order and a sub-graph. Note that the whole graph and its sub-graphs are dynamic and change in the course of interaction that results from collective attempts by agents to find locally and globally beneficial exchanges (see Figures 6.7 and 6.9). The ability to represent different objects and types of link is supported by the property graph model.

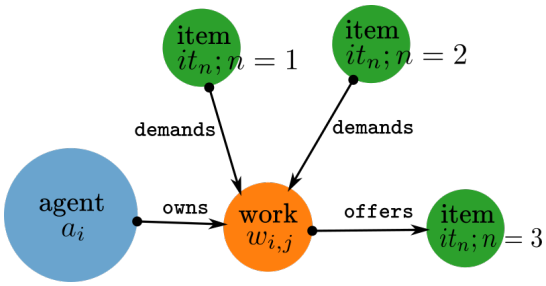


FIGURE 6.4: Graphical representation of preferences in OfferNets in terms of relations between agents, works and items.

From the mathematical perspective, the described OfferNets structure and dynamics can be unambiguously and equally well formalized as an optimization problem (Goertzel, 2015b), a reaction network in artificial chemistry (Dittrich, Ziegler, and Banzhaf, 2001; Heylighen, 2017) or a graph match and search problem (Goertzel, 2017b). However, the mathematical perspective is in a sense "logically centralized" and requires assumptions to be made and global modelling constraints imposed that detach the resulting model from observable dynamics in real-world decentralized markets (not unlike "mainstream" economic thinking). These constraints are best formulated in terms of preferences (Hansson and Grüne-Yanoff, 2018):

i. *Individual (psychological) preferences are known and complete.* This constraint reflects the assumption that agents (humans or AIs): (1) can name their preferences – i.e. make a list – and (2) can unambiguously order this list, at least potentially. We will leave aside the phenomenological issue of whether any real world item is in principle describable by a list of properties that can furthermore be ordered by their "essentiality".

ii. *Revealed preferences are complete.* Revealed preferences are those which an agent discloses to other agents engaged in interaction or a market place. In a sense, revealed preferences are more important than individual psychological preferences in the context of search and match – since preferences have to be externalized in order to facilitate any interaction in the first place. We assume that normally not all psychological preferences are revealed. The correspondence between revealed and psychological preferences is a separate, while not unimportant, question which we again will not ponder here.

iii. *Preferences are consistent.* In the context of OfferNets, consistency of preferences means that they are stable in time. This applies both to individual psychological and revealed preferences, but the latter are more important.

With these constraints in mind, recall the example of a flea market. Obviously, none of the constraints hold in this context: buyers do not know exactly what their preferences are before seeing items, while sellers may change their prices depending on the interaction with a potential buyer. Preferences of either get individuated and only then revealed, and revealed preferences are not complete[15]. Furthermore, no preferences are stable – one can have a completely different "set" of preferences after walking 10 minutes or half an hour in the market. Actually, most of these constraints do not hold even in regular physical or on-line shopping: it is common to

[15]There is an inherent asymmetry among revealed preferences of buyers and sellers, where the latter normally reveal more. This asymmetry has been addressed by *web Of Needs* project (Kleedorfer and Busch, 2013).

emerge from a shop with completely different purchases in the basket than were imagined before entering it. On the other hand, there are different markets where these constraints may hold with different strengths: e.g. standardized commodity, stock or transportation markets, where both supply (offers) and demand can be specified a priori and interaction kept to a minimum. Yet these can be considered rather exceptional, where supply and demand prices carry most of the information needed for decision making. The conceptual question that offer networks poses is therefore whether and how complex exchanges can be enriched and possibly their micro-economic dynamics affected by introducing ways for multi-dimensional information exchange (Goertzel, 2015a). Enabling dynamic interactivity of buyers and sellers is essential for large scale practical applicability of this idea and this is precisely why the offer networks model is well suited for implementing open-ended decentralized computing architecture.

From the computational point of view, OfferNets can be described in terms of a combination of a *data structure* and *processes*. It is customary (and remarkably useful in the practice of designing computational systems) to think of processes as functions operating on data structures by mutating them. Yet it is entirely reasonable (but not equally intuitive) to think of processes without reference to *a priori* data structures which they change – which relates to progressive determination and stigmergic computing (see Section 4.5).

6.3.1 Data structure

OfferNets is defined according to the *property graph model* which: (1) contains nodes and relationships; (2) nodes contain properties and can be labelled; (3) relationships are named, directed, can contain properties and always have start and end nodes (see page 42). This model is more expressive than a theoretical mathematical graph and can represent arbitrarily complex real world structures[16].

OfferNets property graph schema includes vertices of type [*agent, work, item*] and edges of type [*knows, owns, demands, offers, similarity*]:

Agents represent actual participants of the economic exchange – *ON-AI Agents* (see Figure 6.3) – which together form a social network. Every agent in the network is connected to at least one another agent. This gives rise to an important property of the graph: it is necessarily *connected*, i.e. there exists a path via *knows* relation between every pair of agents. This property is crucial for decentralized computing, as will be explained later. Agents also relate to one or more *works* via *owns* links – representing situations in which an agent publishes that it "wishes" to exchange something in the network.

OfferNets is not only a data structure, but also a computational medium and as such has to account for computing resources needed to run decentralized processes. Agents participating in the network (people or AIs) are those entities which own computing resources. Note a direct relation of OfferNets to the actor model

[16] A *hyper-graph*, which may better correspond to many biological and neural structures found in nature (Goertzel, 2017a) is a generalization of the property graph in that it allows one edge to connect more than two vertices. Hypergraph data structures are preferred by some AGI (namely OpenCog) and reasoning (namely GRAKN.AI) systems. Note, however, that a property graph can be transformed to a hypergraph without data loss, but not the other way round in a general case (Blockeel, Witsenburg, and Kok, 2007; Robinson, Weber, and Eifrem, 2015).

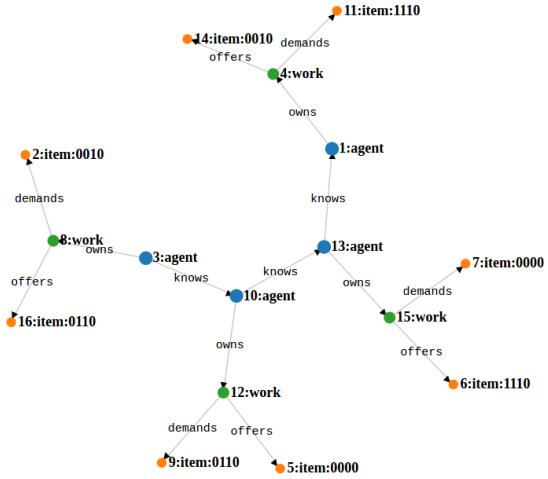


FIGURE 6.5: OfferNets graph structure. Note that this is an initial structure which does not contain *similarity* links, which appear in the graph only after running processes, as explained in Section 6.3.2 on page 160.

of computation (see Section 5.1) where every actor controls its own computing resources. Moreover, note the similarity to the notion of the computation graph as a self-organizing interaction of elementary processes, which makes OfferNets a special case of open-ended decentralized computing (see Figure 5.10). In this sense, *work* is a special *process* (see Section 4.5.1) with *demand* as an input and *offer* as an output. This process is neither immediate nor resource-free: any actual exchange involves costs (time, transportation or other arrangements) which have to be represented in any practical application. These, and similar, issues are left out from the current simplified model of OfferNets, but can be readily implemented by including additional features as defined the by general model of open-ended decentralized computing.

Work therefore represents the "process of exchange" in which an agent is willing to engage with other agents in OfferNets. By "owning" a *work*, an agent pledges that the work will be executed if a match is found[17]. In the current simplified OfferNets model *work* connects only one *demand* with one *offer*, yet in principle arbitrary complex works can be represented featuring more than one input (energy, computational resources or a monetary payment) or output. Moreover, *work* can represent the exchange of data output (e.g. textual description of an image) for data input (e.g. an image to be described), a monetary payment (in fiat or crypto- currency), computational resources needed for the task, or location of the process in the network's topology. In this case, a *work* would not be a "process of exchange" but rather a "process of text summarization" which nevertheless can be perfectly well represented within the same framework.

An **item** is an actual "thing" that is put forward by an agent for a potential exchange. In OfferNets this is limited to actual physical or non-physical things, but in general an item could be a representation of any input or output of a generalized process (i.e. data, token, energy units, etc.).

In order for the processes operating within OfferNets (Section 6.3.2) to be able

[17] Any such pledges in a decentralized system have to be supported by the system of "distributed trust", which is another important and interesting issue which is outside the scope of our present treatment of offer networks.

to find matching offer and demand pairs which could be exchanged, a *similarity* relation between any pair of items should be defined. That is, given an arbitrary pair of items registered in the network, a process or algorithm should be able to calculate how similar (or dissimilar) they are. If and when the similarity between two items is calculated, a symmetric *similarity* relationship between them is created in the OfferNets graph[18].

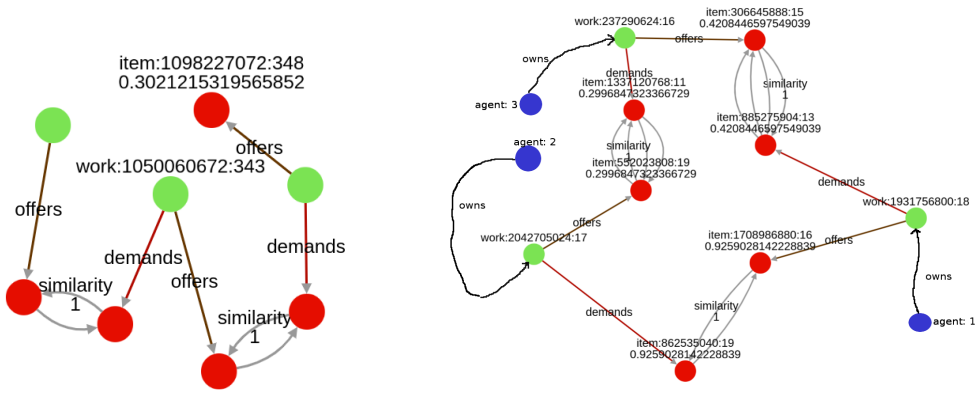
FIGURE 6.6: OfferNets graph schema.

This means that every *item* has to contain a description according to which its similarity with other items in the network can be potentially estimated or calculated. Here we touch again the issue of properties and preferences (see page 156). For the purposes of this OfferNets simulation, the description of an item and similarity relation are defined as a real number between zero and one. While these choices of definitions are not arbitrary, they are of minor importance in the context of experimenting with centralized and decentralized search and match in OfferNets. It should nevertheless be noted that representation of and similarity measuring between real-world items is a large and interesting domain of inquiry in itself. Similarity can be an arbitrary complex relationship depending on the practical application of the framework. A truly decentralized Offer Network type system should be able to accommodate different and possibly competing variants of measures and algorithms. The architecture of OfferNets allows for plugging in any chosen similarity representations.

Relations of similarity between items enables chains and cycles of works to be found. A **chain** forms itself when there is an item i_1 offered by work w_1 and item i_2 demanded by work w_2 and when items i_1 and i_2 are sufficiently similar (see Figure 6.7a). "Sufficiently similar" in this context means that the calculated similarity measure between items is greater than or equal to an arbitrary threshold set by an agent engaging in the exchange of items (different agents may have different thresholds). If this parameter is equal 1, then the requirement is to exchange only strictly similar items, which results in a standardized exchange. A **cycle** is simply a chain forming a loop (see Figure 6.7b). When a chain or a cycle is found in OfferNets, it indicates the possibility of an actual exchange between agents.

Apart from the chains and cycles illustrated in Figure 6.7, works can be connected in arbitrary complex workflows in cases where a domain model allows them to have multiple demands and offers (see Figure 6.8). The problem of search and matching is precisely a generalized process of finding workflows in such a network – the complexity of which as well as a comparison of algorithms from the mathematical perspective are investigated by Goertzel (2017b).

[18] In the OfferNets graph schema, a symmetric *similarity* relation is represented by two directed edges $a \xrightarrow{\text{similarity}} b$, $b \xrightarrow{\text{similarity}} a$, where a, b are items and $a \neq b$



(A) Chain (B) Cycle

FIGURE 6.7: Examples of a chain and a cycle in OfferNets. Red spheres are *items*, green – *works*, blue – *agents*. Figures are visualizations of the results of an OfferNets simulation.

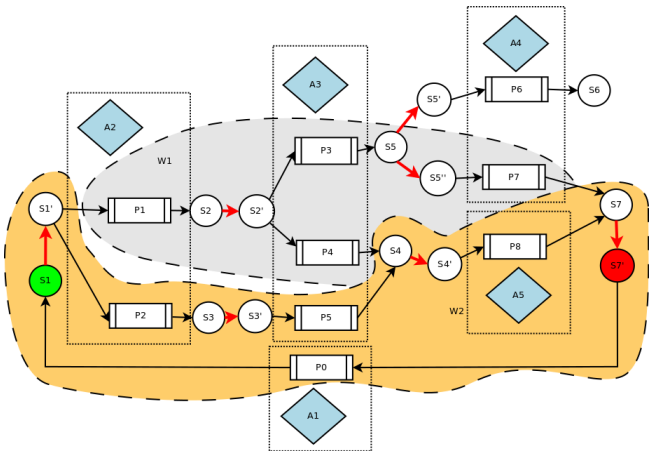


FIGURE 6.8: Data centric approach, describing workflow as a sequence of transitions between data states (inspired by Cushing, 2015; Cushing et al., 2015). In OfferNets terms, *work* here is represented as a process P_i, *items* as data states S_i and *agent*s as blue rectangles A_i.

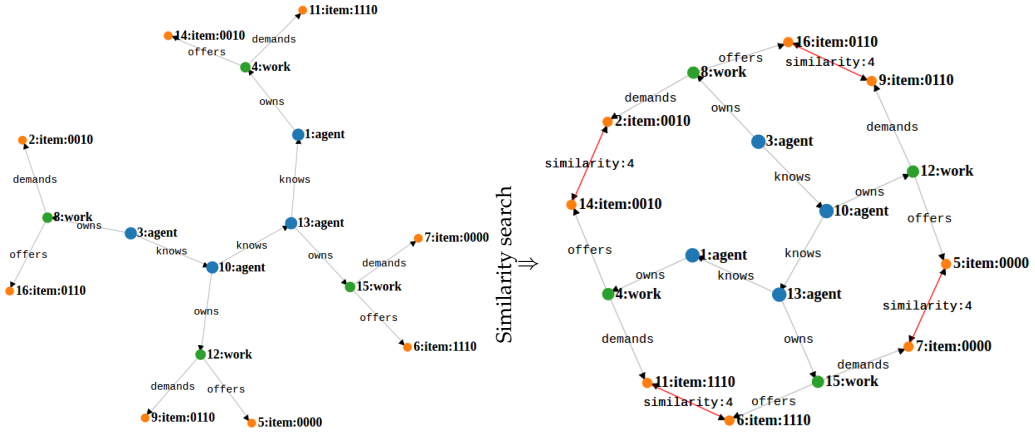
6.3.2 Processes

Following the basic principles of open-ended decentralized computing, OfferNets is implemented as an ecosystem of decentralized processes interacting via a stigmergic medium. The list of decentralized processes is open-ended in the sense that any process can be executed by an agent participating in the network. Processes needed for basic functionality of OfferNets are: (1) similarity search, (2) find cycles of changeable items, (3) execute exchange cycles and (4) find and connect items of exchange via similarity links. These processes are described in detail below.

Process #1: Similarity search

This process searches similar *items* in the network according to given criteria and connects them with explicit *similarity* links. The ability to measure similarity of items

is based on an agreed representation and description of each item. Similarity measures can also take different forms depending on item representation. If the representation is the real number, then the similarity measure could be a difference. If items are described by vectors, similarity can be measured by a Hamming distance[19] or as cosine similarity[20]. In the case of natural language descriptions or images, NLP techniques or image summarization could be used. Note that these measures are global parameters of simulation framework and different forms of them can be "plugged" and "unplugged" dynamically. Furthermore, in a decentralized system, nothing prevents agents from agreeing themselves on the usage of different similarity measures within the same framework. In terms of the overall dynamics of OfferNets, the goal of the similarity search process is to change the topology of the graph so that similar items become topologically closer to each other (see Figure 6.9 and Figure 6.10).



(A) Before similarity search processes: 12 nodes (of type *agent*, *work*, *item*); 12 edges (of type *knows*, *owns* and *demands* | *offers*); zero *similarity* links.

(B) After similarity search processes, 4 *similarity* links forming a cycle.

FIGURE 6.9: Visualization of graph mutations and topology change due to similarity search process and *similarity* links creation on a small OfferNets sub-graph.

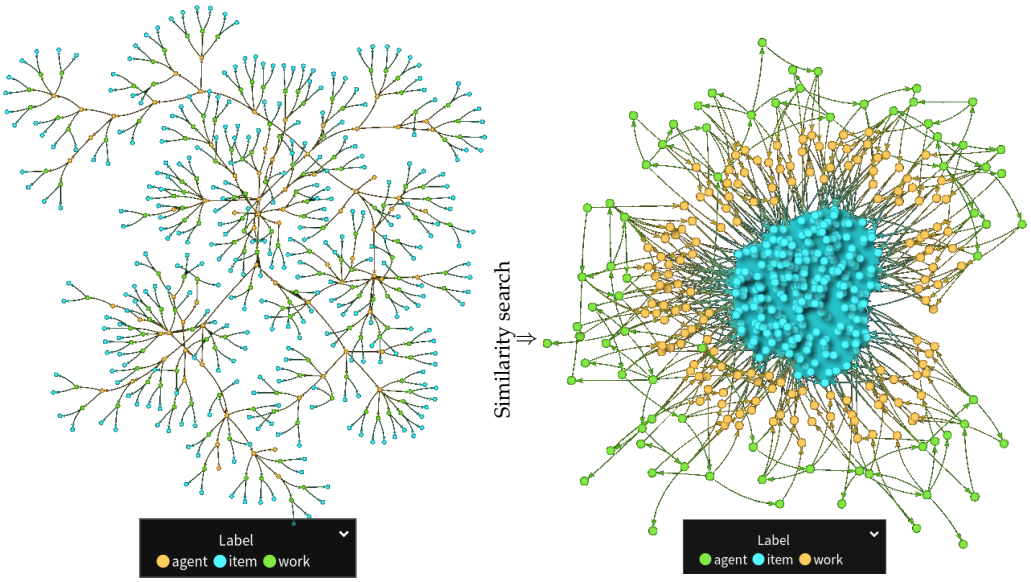
In this example *item* values are represented as real numbers in the range $[0, 1]$. Similarity between two items is then calculated using the formula $Sim = 1 - abs(value_{i1} - value_{i2})$ which also results in the real number in the range $[0, 1]$. The closer this number to one, the more similar the items are.

Algorithmically, the similarity search process is implemented in two ways – centralized and decentralized. The comparison of their performance forms the basis of the experiment discussed in Section 6.4.

Centralized similarity search

A centralized similarity search simply fetches all *items* in the network, compares each of them to every other and creates a *similarity* link among those that have a similarity value exceeding a parameter of *similarityConnectThreshold*. This parameter regulates the density of connectivity between *items* on the one hand and the ability for agents to exchange "fuzzy" similar items on the other.

[19] https://en.wikipedia.org/wiki/Hamming_distance
[20] https://en.wikipedia.org/wiki/Cosine_similarity



(A) Before similarity search processes: 534 nodes (of type *agent*, *work*, *item*); 534 edges (of type *knows*, *owns* and *demands* | *offers*); zero *similarity* links.

(B) After similarity search processes, 46.574 *similarity* links

Nodes		Links	
Agent	75	'knows'	74
Work	153	'owns'	153
Item	306	'demands' / 'offers'	306

FIGURE 6.10: Visualization of graph mutations and topology change – same as Figure 6.9 but on a larger graph.

A centralized similarity search requires a full scan of the graph in order to collect data on all items demanded or offered by agents at a certain moment in time, combining this data into a single data structure and then processing it in a centralized (but possibly distributed) manner.

Decentralized similarity search

A decentralized similarity search, unlike the centralized one, works only on behalf of an agent that initiates the search and does not require the fetching of all *item* values from the network. On the other hand, a decentralized process requires concurrent and asynchronous execution on behalf of each agent. It operates as a spreading activation which starts with the initiating agent's items and checks their similarity to those of the agent's neighbours. A decentralized similarity search takes *similarityConnectThreshold* and *maxDistance* parameters. The former serves in the same way as in a centralized search, while the latter determines how far into an agent's neighbourhood the spreading activation process traverses.

The process follows this logic (see also Listing 6.1 for the pseudo-code):

i. traverse *knows* relations in a *agent* $\xrightarrow{\text{knows}}$ *agent* sub-graph starting with agent a_1 and reach all connected items i_i of a "friend of friend" within distance *maxDistance* from a_1;

ii. calculate a similarity measure $s = similarity(i_1, i_i)$ for all found items and create a similarity relation between i_1 and i_i if and only if the global parameter *similarityConnectThreshold* is larger or equal to s.

LISTING 6.1: Pseudo-code of similarity search process via $agent \xrightarrow{\text{knows}} agent$ sub-graph.

```
# parameters:
# -- me: the agent that initiates traversal
# -- maxDistance: number of hops in traversal;
# -- similarityConnectThreshold: only items with this and higher similarity
    are connected;

    myItems <- me.getAllDemands() + me.getAllOffers();
    discoveredItems <- emptyList();
    distance = 0;

    function getItemsOfNeighbours(agents):
            for each agent in agents do:
                discoveredItems <- discoveredItems + agent.getAllDemands() +
                    agent.getAllOffers();
                neighbours <- agent.knowsAgents();
                getItemsOfNeighbours(neighbours);
                distance = distance +1;
                if distance = maxDistance do:
                    break from cycle;

    getItemsOfNeighbours(me)

    for each discoveredItem in discoveredItems do:
            for each myItem in myItems do:
                similarityValue = calculateSimilarity(discoveredItem, myItem)
                if similarityValue >= similarityConnectThershold do:
                    createLink(from: myItem, to: disoveredItem, type:
                        similarity, value: similarityValue)
```

Running this process a sufficient number of times results in items with similar values forming densely connected clusters in the graph – i.e. it alters network topology as illustrated in Figure 6.10 in a way that enables decentralized search of paths and cycles as well as making further similarity searches more efficient.

Process #2: Find cycles of changeable items

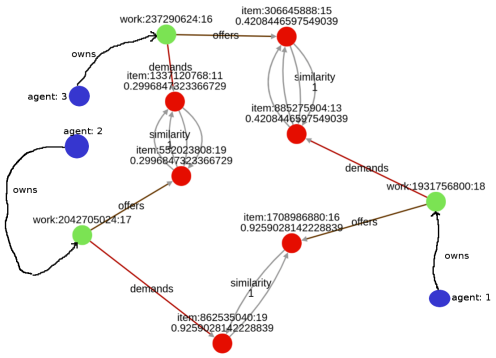
In graph theory, a cycle is a collection of vertices and edges between them where each vertex is reachable from itself via edges present in the collection (see Figure 6.7b). Cycle search is the process that finds such data structures in a messy and unstructured initial graph. Concretely, in OfferNets, a decentralized cycle search process is a *vertex centric graph traversal* which, for a chosen work w_1 owned by an agent a_1, and given *similaritySearchThreshold* and *maxDistance* parameters, initiates the following process:

i. traverse *offers* and *demands* relations starting with work w_1 and find sub-graphs of the pattern $work_1 \xrightarrow{demands|offers} item_1 \rightarrow similarity_{12} \rightarrow item_2 \rightarrow work_2$ (see Figure 6.7a), where $similarity_{12} >= similaritySearchThreshold$; repeat the same traversal until encountering the start of the traversal $work_1$, but not more than *maxDistance* times;

ii. return information on paths and cycles found to $agent_1$, which initiated the traversal.

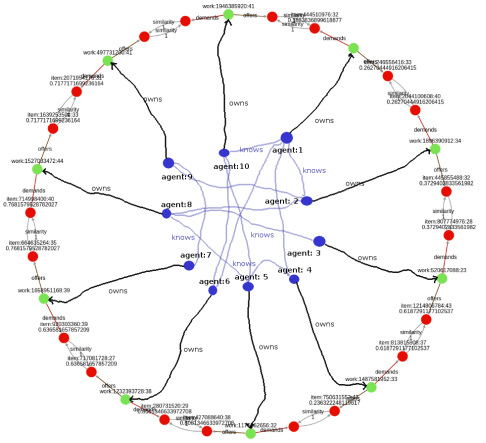
A discovered cycle (see Figure 6.11 below) represents a match of demands and offers of at least two agents participating in the marketplace and the possibility of actual exchange between them. More formally, it is a sub-graph of the OfferNets, where agents "know" other agents' preferences with respect to the discovered match

and can agree to execute the exchange cycle (see process #3). Importantly, the cycle in this case is discovered solely via the decentralized vertex-centric graph traversal without any central processing or data storage – as required by the model of open-ended decentralized computing.



(A) Small cycle

Agents	3
Works	3
Items	6



(B) Larger cycle

Agents	10
Works	10
Items	20

FIGURE 6.11: Cycles discovered in the OfferNets graph by cycle search processes; note how it relates to the conceptual architecture of offer networks (Figure 6.3 on page 155).

Cycles are temporary structures – they get dissolved when executed. Actually, the OfferNets system can be considered more successful and beneficial when more cycles are discovered, executed and dissolved per unit of time, which is a measure of fluidity of the network. Another important aspect, emphasized in the figure, is that while cycles are of dynamic nature – emerging and dissolving during operation of the OfferNets graph – they are basic elements of the conceptual architecture of the self-organizing network of interacting agents (see Figure 6.3).

Process #3: Execute exchange cycles

Finding cycles only increases the probability that certain items will be exchanged, but is far from guaranteeing it. While in a centralized system an execution of the discovered cycle is straightforward, it is not so in the decentralized case. First, since many search and match processes are executed asynchronously, more than one cycle can emerge involving overlaps of agents and items. In this case, a consensus has to be reached, via negotiations, on which cycle will be executed and which will be neglected. This already involves a game-theoretic aspect. Second, when a cycle involves fuzzy matches, the willingness of agents to exchange items that are not strictly similar has to be confirmed. Third, the presence of insufficient information in the network due to incomplete preferences (Mandler, 2005) should be considered – implying the necessity of a round of negotiations between agents involved in the cycle before its execution. Finally, exchanges in distributed systems cannot rely on a single provider of trust and therefore must use a distributed trust model (Abdul-Rahman and Hailes, 1997) including, but not necessarily limited to, the blockchain technology (Swan, 2015) and a reputation system or systems (Kolonin et al., 2018). When an agent receives a list of paths and cycles from the cycle search process, it

decides, according to an built-in logic (which may be different for heterogeneous agents), whether to initiate a transaction in case a valid cycle is found. The resulting dynamics of an OfferNets depends on competition and collaboration between agents with different logic and goals.

Process #4: Search and connect similar items of exchange via similarity links

Computational complexity and success in finding matches in the graph are mostly due to the similarity search process rather than the cycle search. Likewise, search via the *agent* $\xrightarrow{\text{knows}}$ *agent* sub-graph is relatively costly because it requires many agents in the neighbourhood graph to be checked in order to find a similar item. Furthermore, in a decentralized scenario and depending on the topology of the graph, the search may not succeed even in cases where a cycle exists. Therefore, it may be much more efficient to traverse the *item* $\xleftrightarrow{\text{similarity}}$ *item* sub-graph directly where similar items are clustered together due to traces of process #1. This type of search would utilize graph topologies on the right rather than left of Figures 6.9b and 6.10b – where similar items are topologically closer, which implies shorter average traversal paths. Listing 6.2 provides a pseudo-code of the logic of this process.

LISTING 6.2: Pseudo-code of similarity search process via *item* $\xleftrightarrow{\text{similarity}}$ *item* sub-graph.

```
# parameters:
# -- me: the agent that initiates traversal
# -- maxDistance: number of hops in traversal;
# -- similarityConnectThreshold: only items with this and higher similarity
    are connected;
# function getItemsOfNeighbours(agent) is defined in pseudocode for Process
    #1;

    myItems <- me.getAllDemands() + me.getAllOffers();
    distance = 0;

    function searchAndConnectViaSimilarityLinks(myItem, otherItem):
        similarityValue <- calculateSimilarity(item, otherItem)
        if similarityValue >= similarityConnectThreshold do:
            createLink(from: myItem, to: iterItem, type: similarity, value:
                similarityValue)
        if distance > maxDistance do:
            break from the loop;
        else:
            nextMostSimilarItem <- item.getSimilarItems(max(similarityValue))
            searchAndConnectViaSimilarityLinks(myItem, nextMostSimilarItem)


    for each myItem in myItems do:
        distance = distance + 1;
        if distance = maxDistance do:
            break from cycle;
        if item.hasSimilarityLinks() do:
            mostSimilarItem <- item.getSimilarItems(max(similarityValue))
            searchAndConnectViaSimilarityLinks(myItem, mostSimilarItem)
        else:
            itemsOfNeighbours <- getItemsOfNeighbours(me)
            for each discoveredItem in itemsOfNeighbours do:
                searchAndConnectViaSimilarityLinks(myItem, discoveredItem)
```

As specified, the above process traverses *item* $\xleftrightarrow{\text{similarity}}$ *item* sub-graph directly if it exists; otherwise, it reverts to traversing the *agent* $\xrightarrow{\text{knows}}$ *agent* sub-graph as defined by process #1. When many such processes operate concurrently, initiated and sustained by different agents, each process changes the local graph structure around

itself, which is then traversed by many "neighbouring" and possibly some "distant" processes. Together, the combination of asynchronous processes and the graph as a stigmergic medium leads to the fluid dynamics of progressive determination (see Section 3.2.4) and open-ended decentralized computing (see Section 4.4.1).

6.3.3 Research questions

The ambition of the OfferNets simulation modelling project is to *conceive, implement and test mechanisms of search, matching and execution of exchange of goods and services in a radically decentralized system*. This ambition is pursued by asking (and answering) concrete research questions which allow the clear formulation and testing of scientific hypotheses – a process leading to incremental building of the system. The current research horizon encompasses the following research questions:

i. *What are the parameters that determine the advantages and disadvantages of decentralized and centralized search algorithms in different contexts*? In principle, OfferNets logic and goals can be achieved by either centralized (global) or decentralized (local) processes running on the same data structure as defined earlier. In practice, however, the feasibility of any of the approaches is largely determined by concrete circumstances and context-specific aspects. For example, centralized algorithms can optimize results at the cost of larger computational complexity needed to carry them, together with the need for central storage of data structure and privileged access to it (see the fundamental trade-off of decentralized computing on page 31). Decentralized algorithms on the other hand may achieve sub-optimal, but still "good enough" results faster at the cost of giving up control of the whole data structure. Note that the very term "good enough" implies context dependency. In order to provide at least some insights to this question we set up centralized and decentralized search processes in OfferNets and tested them with different parameters (see Section 6.4 for design and results of a computational experiment on comparing decentralized versus centralized search).

ii. *Can we conceive processes which make themselves more efficient by utilizing results of "traces" left by preceding processes*? Such processes, interacting with each other in a progressively determining way, would implement the learning ability of the network. Similar to the first research question, this can be implemented in a centralized or decentralized manner.

6.4 Centralized versus decentralized processes of OfferNets

This section presents and discusses results of the computational experiment designed to answer the research question concerning the advantages and disadvantages of decentralized versus centralized similarity search process in OfferNets. It demonstrates how principles, including, but not limited to stigmergic cooperation between software agents, can be utilized in a software architecture based on the open-ended decentralized computing principles.

6.4.1 Setup

In order to compare process #1 and process #2 in their both centralized and decentralized implementations we follow these steps:

1. first, we create an OfferNets graph of predefined size. We experiment with the random graph (where agents are randomly connected with *knows* links) and a small-world graph, where we know the diameter of the network in advance;
2. then we artificially create a list of items which, if correctly searched and connected in the OfferNets graph, will form a chain (see Figure 6.7a). The length of the predefined chain is set by the simulation parameter *chainLength*;
3. items from the list are assigned to random agents in the network;
4. then, we create a special *taskAgent* owning a *work* which, when correctly connected to the potential chain inserted into the network by step 2, closes it into a loop forming a cycle (see Figure 6.11);
5. finally, we run the decentralized and centralized processes on the same graph and record the running times of each method.
    - The similarity search process connects all similar items with *similarity* links, as explained in Section 6.3.2;
    - The cycle search process is run on behalf of *taskAgent* and discovers the cycle inserted by step 2 only if the similarity search process connects a large enough number of items – as explained in Section 6.3.2.

What is hereinafter called an "experiment run" is a series of simulations, each of which takes a different combination of the following parameters for the purpose of exploring the parameter space (see Figure 6.12):

- *agentNumber*: the number of agents in the network;
- *similarityConnectThreshold*: the minimum similarity value between items connected with explicit link by similarity search process;
- *chainLength*: the length of an artificially created chain inserted into the network by step 2;
- *similaritySearchThreshold*: minimal similarity of items to be considered as eligible for exchange;
- *maxDistance*: radius of agent's neighbour network when searching for similar items;
- *randomWorksNumberMultiplier*: the number of random works and items which are assigned to the agents in the network to make cycle search more realistic;

6.4.2 Observed dynamics

Here we present the observed dynamics of simulations, mostly in terms of the relation of time to the topology and size of the OfferNets graph. We briefly describe the technical aspects behind these observations. Interpretation of and conceptual insights from the observations are discussed further in Section 6.5.

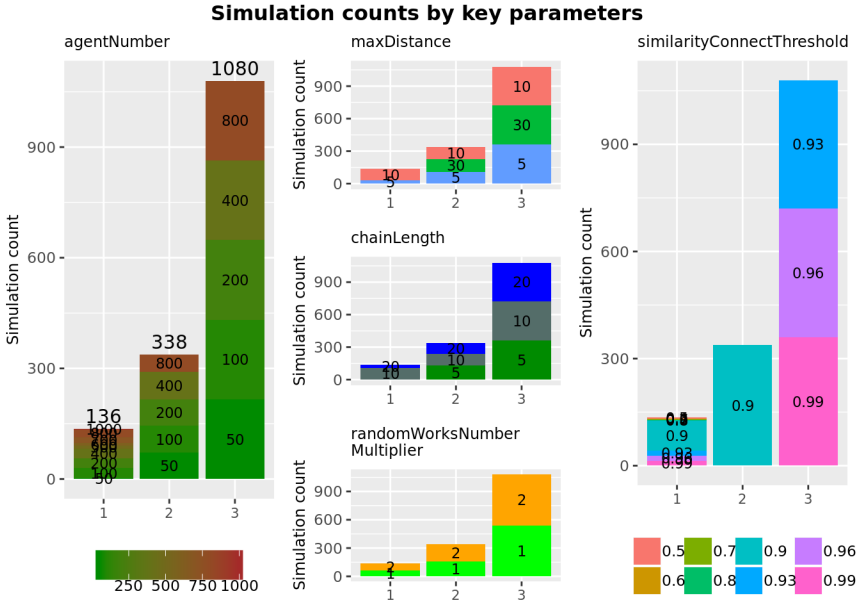


FIGURE 6.12: Distribution of simulation parameters of all analysed experiments (total number of simulations: 1,554; aggregate simulation time (user): 757 hours, events cached: about 1 billion).

Sensitivity to graph topology

First of all, data collected from simulations supports the intuition that decentralized and centralized searches have very different sensibility to the underlying graph topology. That is, the centralized search algorithm is more sensitive to how many agent nodes are in the graph rather than on how well they are connected. The decentralized search, on the other hand, is sensitive to the topology of the *agent* $\xrightarrow{\text{knows}}$ *agent* sub-graph. In part this is because decentralized graph traversals continue only as deeply as constrained by *maxDistance* parameters, which define the radius of exploration (see Section 6.4.1). Due to this, where the diameter of the graph is larger than *maxDistance*, the cycle may not be found, even though it exists in the network. It is of course possible to increase the *maxDistance* parameter to the arbitrarily large number, but this also may increase the time (see Figure 6.13). Another aspect is that the computational complexity of deep graph traversals depends on how many outgoing links every traversed node has (i.e. the branching factor of the tree constructed by a traversal) – which is a parameter of network topology. Obviously, the larger the branching factor the more computationally complex it is to traverse it.

For confirming the sensitivity of the decentralized search to graph topology we also ran simulations with the same parameters on two different graph topologies – (1) randomly connected (Erdös and Rényi, 1959) and (2) small world (Watts and Strogatz, 1998), with a diameter of less than 10. As shown in Figure 6.14, the success rate of finding a cycle in a random graph is often lower than 100%, except when the traversal depth is made very large – which takes more time, as shown in Figure 6.13. On the other hand, a small world graph structure with a known diameter and corresponding *maxDistance* parameter guarantees that the cycle will be found if it exists in the graph, even in a decentralized scenario.

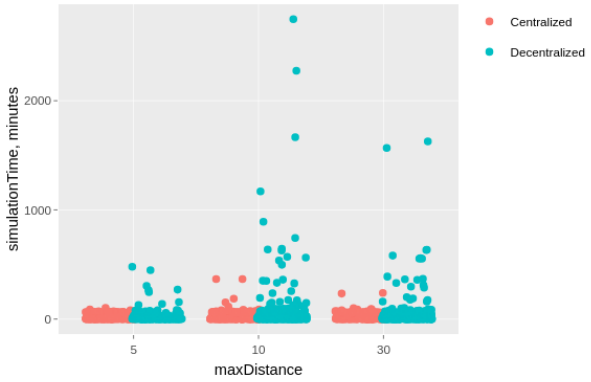


FIGURE 6.13: Dependence of decentralized search time on *maxDistance* parameter. Traversals with higher *maxDistance* have a higher probability of running longer. Runtimes are not deterministic because they depend to some extent on fine graph topology (i.e. $agent \xrightarrow{\text{knows}} agent$ sub-graph which is generated randomly).

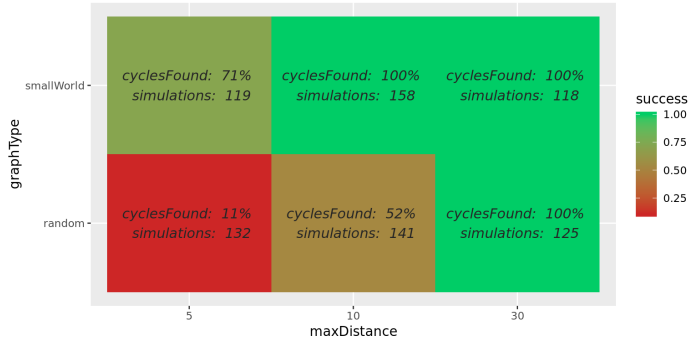


FIGURE 6.14: Success rate of finding a cycle in random and small world graphs with decentralized traversals with different *maxDistance* parameter. Small world graphs have a known diameter of *10*. Note that with random graphs, *maxDistance* of *5* and *10* is not enough to traverse the whole graph, while *30* often provides (but does not guarantee!) full coverage. Likewise, a decentralized search with $maxDistance = 5$ does not always find a cycle in the graph with diameter *10*.

Decentralized search time is sensitive to the number of edges

Figure 6.15 shows the dependency of simulation time on the number of *similarity* edges in OfferNet graphs with a different number of agents. The decentralized search seems to be marginally faster when there is relatively small number of links in the graph, yet its time complexity quickly exceeds the centralized search when the number of similarity edges increases.

This data show that the centralized search is almost always faster than the decentralized. Note also that the decentralized search may find the same cycle several times during the same simulation, which is not possible in the centralized case. This is because similarity search and cycle search processes run asynchronously and independently of each other. Therefore there is no easy way to stop all processes when one of them finds the cycle.

Centralized search time increases with the number of vertices

The dependency of time of simulation on the number of total vertices in the graph (i.e. of type *agent*, *work* and *item*) is visualized in Figure 6.16.

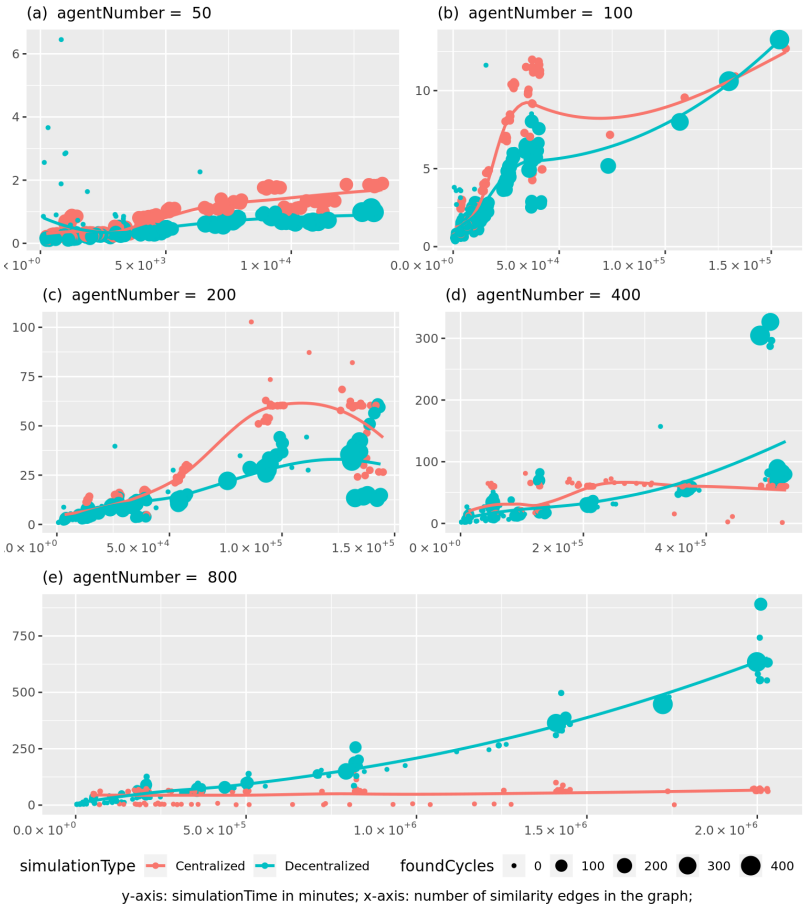


FIGURE 6.15: Dependence of simulation time on the number of *similarity* edges in the graph.

Not unexpectedly, the time needed for the search increases with the number of vertices in the graph, which follows common sense intuition. This is correct for both centralized and decentralized versions. Note, however, that in decentralized traversals it is difficult to separate the role of number of vertices from the role of number of links, since they are highly correlated variables. Nevertheless, we can at least approximately estimate the effect of the number of links in OfferNets simulations with the help of the *similarityConnectThreshold* parameter. Recall that this parameter sets the minimum value for items' similarity in the network for them to be connected with a similarity link. Figure 6.17 shows how it modulates the topology of the graph, since when the parameter is lower, the probability of two items getting connected is higher, and therefore total number of edges in the graph is higher given the same number of vertices.

As can be seen from Figure 6.16, simulation times in the centralized search are almost non-dependent on the magnitude of the *similarityConnectThreshold* parameter. The opposite is true for the decentralized search, where the spread between lines representing different *similarityConnectThreshold* is obvious. Therefore, the centralized search is more sensitive to the number of vertices, and the decentralized to both number of edges and vertices.

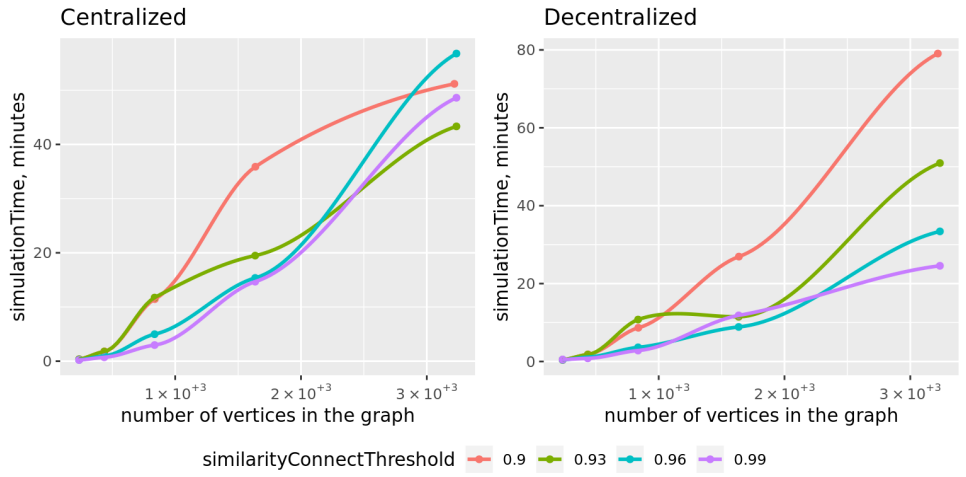
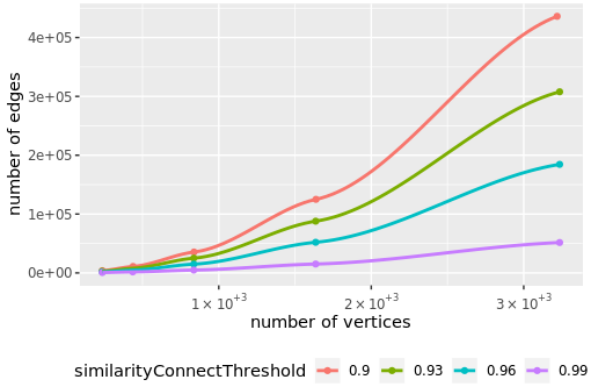


FIGURE 6.16: Time needed for centralized search roughly increases depending on number of vertices in the graph.

FIGURE 6.17: Graph topology as modulated by *similarityConnectThreshold* parameter. The lower value of this parameter on average means that more *items* get connected with *similarity* links, and therefore the total number of links in the graph increases.



6.5 Summary and discussion

This chapter presents and discusses the implementation of selected aspects of a decentralized exchange using the software architecture of open-ended decentralized computing. Concretely, we have implemented the logic of the offer networks domain model on the simulation engine, run a number of simulations with different parameters and presented their results with the help of analysis and the monitoring engine designed in Chapter 5. Within the scope of this work, the goal of OfferNets simulation modelling experiments was, first of all, to gain insights from actual implementation and operation of the architecture in the running computer code. Recall that the design-based research methodology of this work is motivated by the desire to connect ultimately abstract with the most specific. This connection is not to be understood as a "research funnel"[21], which describes stages of research in terms of how close they are to the problem that is being addressed by a research project, where the sole purpose of the funnel is to direct and restrict all stages of research towards the *a priori* formulated problem. In our case, philosophical, computational, software design and practical implementation are all aspects that inform each other and the whole research programme. The implementation and application of the framework to practical problems in a well defined domain adds a pragmatic layer

[21] Boulton, E. (2014, July 22). The Research Funnel [Blog]. Retrieved January 17, 2019.

to approaching the problematic of open-ended computing and dealing with decentralized complex systems as well as the centralization – decentralization dilemma in general.

Open-ended decentralized computing deals with *uncertainty*, rather than *efficiency*, which is a primary concern of closed computing (see Section 4.3). Simulation modelling experiments with OfferNets clearly expose this distinction. Observe, from the comparison of centralized and decentralized search results in Section 6.4.2, that the centralized search is more efficient in almost all cases. Leaving aside (without neglecting) the specifics of technical implementation of both types of algorithm, observe that the design of research questions for the computational experiments follows the hypothetico-deductive method of science[22]. In our case this method implies formulating a goal and then comparing how well centralized and decentralized algorithms fare in reaching it. This approach is perfectly legitimate in pragmatic terms, yet also misses the point – because the very existence of a common goal is set to measure efficiency, yet decentralized computing and algorithms deal with uncertainty instead. Therefore, the two are not easily comparable. Obviously, a computing model that deals with efficiency (i.e. centralized) will fare better when presented with a task requiring efficiency. However, this does not say that there is no point in comparing the two approaches – but that a just comparison has to involve both efficiency tasks and uncertainty tasks and the latter are more difficult to design, if at all possible within the framework of the hypotetico-deductive method.

Furthermore, observe that setting a goal for an algorithm in computational terms means setting a halting criterion for a computational process in the Turing (1937) sense. However, open-ended computing is conceived specifically for being able to define computation which is not dependent on explicit halting criteria. In OfferNets experiments it is revealed by the following: while a centralized search can be performed and then checked for correctness once, after the algorithm has been halted, a decentralized search has to check the cycle and path found by every asynchronous process internally and then halt when one is found – which clearly is more computationally costly in a goal-directed setting. Conceptually, this insight points to the observation that self-organization, open-endedness and open computation are more costly computationally and in terms of resources than goal-directed processes and closed computation when approached from the perspective of their final product. Consider, for example, biological evolution. The absolute majority of species that ever lived on Earth are extinct. Moreover, most of the evolutionary variations (a sort of decentralized exploration) are unsuccessful. If looking from the perspective of "final products" – i.e. currently living species – evolution is a huge waste of resources. Yet, if looking from the starting point, where nothing is determined, the explorations that supposedly lead nowhere are necessary for "organizing the unknown". This metaphor applies also to AI research perspectives (see Section 2.1) and above all to the distinction between AI and AGI. In the same manner, if the goal of an AI engine, machine learning implementation or a robot is known in advance, it will be always more computationally efficient to design a concrete algorithm which reaches that goal. Yet if one is concerned about intelligence expansion, radical novelty and the diversity of its forms, one is necessarily faced with the issue of dealing with uncertainty, which no combination of goal-directed AI engines or processes can account for. For one, this allows us to appreciate the power of the computational metaphor.

[22] See Section 1.3 for discussion of the hypothetico-deductive method of science.

Moreover, depending on the level of self-organization, surprise, generality and ability to deal with uncertainty that is desired from our systems – computational, social, economic or socio-technical– we need to balance efficiency criteria with the additional resources necessary for self-organization. The conclusion that determinism is a special case of non-determinism (see Section 4.2.3) is equivalent to the statement that goal-directed intelligence and behaviour is a special case of open-ended intelligence (see Section 2.2.3). In summary, self-organization itself is computationally costly and may be not the most efficient way of achieving results, when the goal of computation is known. On the other hand, computational self-organization is necessary when the problem domain is not clear.

The overall role of this chapter in terms of implementation of the offer networks domain model on the basis of open-ended decentralized computing architecture is to demonstrate the importance of connecting the deep philosophical and conceptual framework to concrete software development decisions. We believe that such connections play a crucial role in complex software development programmes, having long term and broad implications, certainly related to the artificial intelligence research programme, but also to complex engineering undertakings in general. We may call this approach *philosophical engineering* – not in the sense of engineering philosophy, but in the sense of emphasizing the role of philosophy *for* engineering complex systems. The next chapter presents and discusses how the open-ended decentralized computing model, and the conceptual paradigmatic shift that it carries, offers pragmatic perspectives to some of the most complex challenges of developing and governing contemporary socio-technological systems.

Chapter 7

Future avenues of application

Parts of the chapter are based on the following working papers and technical reports:

Veitas, V., Heylighen, F., Hodne, T.-E., Lenartowicz, M., Weinbaum, D. R., and Beigi, S. (2015). Governing the Future's Power System: toward a method of guided self-organisation for the Smart Grid. Working Paper. Brussels: Global Brain Institute, CLEA-VUB.

Veitas, V. K., & Delaere, S. (2018). Policy Scan and Technology Strategy Design methodology (Technical report) (p. 20). Brussels: imec-SMIT-VUB.

Veitas, V. K., & Delaere, S. (2018). In-vehicle data recording, storage and access management in autonomous vehicles (Technical report) (p. 21). Brussels: imec-SMIT-VUB.

7.1 Open machines

At the beginning of this thesis we provided a broad introduction to the modern quest for designing synthetic intelligence. Over the history of the last seventy years of this quest, many paradigms, branches and roots of conceptual, fundamental and applied thinking have been merged and also divided. Some of the branches of this quest have explicitly accounted for the desire to understand intelligence at large, others have contributed to it implicitly, while others have championed a clear-cut "no-nonsense" technical solution building. Yet some other branches I have left out entirely or under-represented, without any disregard for their importance – only in an attempt to keep the path of the design inquiry in check.

I look at all aspects of AI research as different schools of thought in understanding what intelligence is at large – natural, human, synthetic or a combination – and, furthermore, how individual intelligences relate, bearing in mind that no intelligence can truly exist individually. Framing intelligence in terms of a technical problem – without downplaying the importance of the amazingly complex technical aspects and solutions that it requires, and even when adopting a strictly pragmatic approach – is neither sufficient nor adequate in the midst of the third "AI spring"[1]. Paradoxically, the insufficiency of purely technical approaches comes to light with largely technical breakthroughs in AI and robotics, bringing autonomous

[1] Bughin, Hazan. (2017, August 21). The new spring of artificial intelligence: A few early economies. In VOX. Retrieved December 14, 2018.

robots, intelligent services, autonomous vehicles and the like closer to everyday life and support systems of human society. Formerly distant questions, previously tackled only by futurists, philosophers of mind and "die-hard" AI researchers, of what happens to human society, understanding of self and universe, when synthetic intelligence comes close to or surpasses the human level (Goertzel, 2014; Kahrs, 2017; Nilsson, 2005), gain importance and increasingly manifest themselves in public debate, mainstream culture and policy decisions. For example, by the end of 2018, the main economies of the world had recognized the magnitude of the socio-economic impact of AI development and adoption, and announced their strategies for governing AI research, industrial policy and legal infrastructure[2]. Most of these strategies, apart from caring about national competitiveness and preparing for socio-economic changes, also consider ensuring ethical and legal frameworks. It is increasingly realized that AI strategies and policies are directly associated to understanding and defining intelligent behaviour that is not solely human-embodied. Consequently, the issues of legal person-hood of artificial intelligence and liability of distributed intelligences (embodied and virtual alike) have repositioned themselves from the domain of purely academic research (Karnow, 1996; Solum, 1991) to pressing social governance matter and legal debate (Cath, 2018; Pagallo, 2018; Solaiman, 2017). This emphasizes the importance of conceptual aspects of seemingly technical questions, which in some cases go as far as proposals for establishing a notion of electronic personhood for advanced autonomous robots (European Parliament, 2017). I believe that the depth of how these issues are approached in the public debate, in social, economic, and technological governance, in education and, finally, in design and engineering, is of utmost importance for setting the direction of the long-term socio-technological development of human society.

Considering the level of embeddedness of AI systems – embodied, virtual and decentralized alike – in society, they are no longer only technological, if they ever were, but socio-technological and economic. The socio-economic repercussions of disruptive technological developments, as much as they are important, are only one aspect. Another aspect is that operation, usage and further development of the socio-technological is interactive, evolving and open-ended, which becomes more important with the increasing autonomy of technical objects. The underlying dynamics of our socio-technological systems is no longer based on creating tools that merely replace or enhance human capabilities, but on those that have or develop capabilities substantially different from ours. In order for such technologies to be continuously integrated into the fabric of society, influence and be shaped by it, it should constitute an integral part of the rich ecology of participatory sense-making, where humans interact with humans, technologies interact with technologies, and technologies interact with humans. Such ecology is best approached in the light of the actor-network theory of Latour (2007) and philosophy of individuation and technology by Simondon (1980, 1992) in terms of a genesis of fluid individuals of hybrid nature and the framework of scalable individuation of social assemblages composed of humans and technical objects.

The following extended quote from Gilbert Simondon's complementary doctoral thesis on the mode and existence of technical objects describes well the significance of the approach to intelligence as the totality of individuating interactions among a population of partially determined fluid natural and artificial identities:

[2]Dutton, T. (2018, June 28). An Overview of National AI Strategies [Blog Post]. Retrieved January 18, 2019.

> Culture behaves towards the technical object much in the same way as a man caught up in primitive xenophobia behaves towards a stranger. This kind of misoneism directed against machines does not so much represent a hatred of the new as a refusal to come to terms with an unfamiliar reality. [... But] culture fails to take into account that in technical reality there is a human reality, and that, if it is fully to play its role, culture must come to terms with technical entities as part of its body of knowledge and values. Recognition of the modes of existence of technical objects must be the result of philosophic consideration; what philosophy has to achieve in this respect is analogous to what the abolition of slavery achieved in affirming the worth of the individual human being. [...] Idolators of the machine generally assume that the degree of perfection of a machine is directly proportional to the degree of automatism. Going beyond what can be learnt from experience, they suppose that an increase in and improvement of automatism would lead to the bringing into oneness and mutual interconnection of all machines – the creating of a machine made up of all machines. [...] Automatism, and that use of it in the form of industrial organisation which we call automation, has an economic or social, rather than a technical, significance. [...] The real perfecting of machines, which we can say raises the level of technicality, has nothing to do with an increase in automatism, but, on the contrary, relates to the fact that the functioning of the machine *conceals a certain margin of indetermination*. It is such a margin that allows for the machine's sensitivity to outside information. It is this sensitivity to information on the part of machines, much more than any increase in automatism that makes possible a technical ensemble. A purely automatic machine completely *closed* in on itself in a predetermined operation could only give summary results. The machine with superior technicality is an *open machine*, and the ensemble of open machines assumes man as permanent organizer and as a living interpreter of the interrelationships of machines (Simondon, 1980, p. 3-4)[3].

Open-ended decentralized computing is a computational model for engineering open machines and their assemblages. In this model the element of indeterminacy, described by Simondon (1980), is expressed in terms of non-deterministic computation. It allows open machines to be sensitive to uncertainty and irregularities which are then progressively solved via interactions within their milieu with other natural, synthetic intelligences and technical objects. This ability of machines and technical objects to resolve indeterminacy via interactions becomes crucially important in the domains of socio-technological development where engineered systems enjoy increasing levels of automation and autonomy. The following section discusses domains of socio-technological development where the open-ended decentralized computing model is most applicable.

7.2 Prospective domains of application

7.2.1 Smart mobility and cooperative intelligent transportation systems

One of the fastest developing areas in the domain of autonomous robotics and artificial intelligence with a potential for large-scale deployment in the short or medium term is that of self-driving technologies and smart mobility systems. The transition, already begun, of conventional transport systems and infrastructure to collaborative intelligent transportation systems, based on self-driving technologies, ubiquitous connectivity and traffic management with minimal human intervention, promises the first social integration of advanced AI of such magnitude (Veitas and Delaere,

[3] Italics added by me.

2018a). The development of the technology of the connected car emphasizes particularly well the pragmatic value and applicability of open-ended intelligence philosophy and the open-ended decentralized computing model.

First, consider an *autonomous car* (see Figure 7.1a). This is a car able to make autonomous decisions based on the immediate data collected from a number of in-built sensors (cameras, Lidar, GPS units) which replace and enhance human drivers' skills and senses. A *connected car* (Figure 7.1b) is the notion of a road vehicle which uses electronic communication technologies for communicating with the driver, other cars and their drivers, pedestrians, road, city, global and local informational infrastructure, traffic managers and more. Such vehicles, including individual or shared cars, shuttles, buses and trams, are envisioned to be able to make collective driving decisions informed by the broad context of the traffic situation at different distances around them. These decisions are made by integrating data not only from internal sensors, but also from the driving decisions of other vehicles, congestion situations on potential travel trajectories, meteorological conditions as informed by "smart city" sensors and many more. Obviously, the vision of a full-blown connected car is closely related to the promise of the internet of things – a global network of interconnected cyber-physical devices. An autonomous car, which is also connected, becomes a *connected autonomous car*.

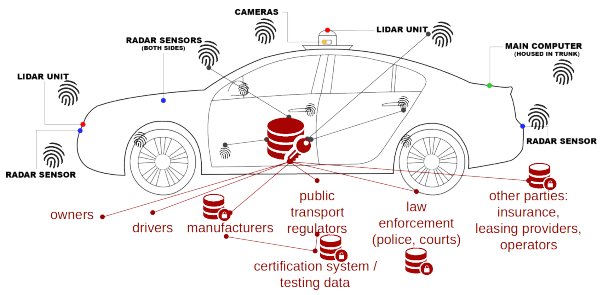
The notion of the collaborative intelligent transportation system (C-ITS) denotes an integrated yet fluid system of connected autonomous cars communicating with each other, road infrastructure, and other C-ITS systems. One of the features of C-ITS is *cooperative awareness*, which refers to automated driving scenarios where a vehicle's local decisions are based on the integration of global data and intelligence (Veitas and Delaere, 2018a). Note that, on the conceptual level, C-ITS is a clear case of distributed intelligence (see Section 2.1.3), an approach that is not unknown in transportation systems research informed by self-organization and stigmergy (Gershenson, 2011). The magnitude of prospective C-ITS' economic and social benefits is crucially dependent not only on automation and cooperation between individual vehicles, but – even more so – on effective vehicle sharing schemes and integration of different transport modalities – taxi pools, on-demand passenger cars, road, water and air transportation (Veitas and Delaere, 2018b). The user-centric perspective to intelligent transportation systems is captured by the notion of *smart mobility*, which refers to a system that offers integrated yet personalized "mobility as a service". Ideally, a smart mobility system should be able to offer the user the best local transportation service, with a trajectory and single payment system which integrate all available modalities and vendors. Since the transport sector accounts for a considerable share of a country's economy and intelligent transportation systems promise to solve many of the current inefficiencies[4], a number of large scale C-ITS and smart mobility projects are close to fruition in many countries[5].

Let us now consider at a more conceptual level the difference between an autonomous and connected car. Regular modern cars are controlled by a large stack

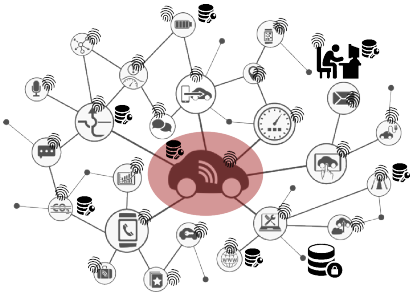
[4]For example, it is estimated, that currently, an individual vehicle is used on average about 5% of time, while the rest of the time it occupies a parking space. An increase of this percentage via smart mobility solutions would greatly reduce the need for private and public investment into the transportation system, as well reducing the ecological footprint of the whole ecosystem of industries.

[5]The start of large scale deployment of collaborative intelligent transport systems is targeted for the year 2019 in the European Union; standardization initiatives and pilot projects are actively being carried out in US, Japan, Singapore, South Korea, China and New Zealand (Weeratunga and Somers, 2015).

of distributed software that executes more than 50 interconnected electronic control units (ECUs) – devices responsible for overseeing, regulating and altering the operation of the different subsystems of a vehicle (Van Bulck, Muhlberg, and Piessens, 2017). Autonomous vehicles may contain hundreds of such nodes, communications between which determine the functionality, automation level and behaviour of a vehicle. Connected cars will operate by integrating the information from *internal* nodes – from inside a vehicle – and *external* nodes – from other vehicles and IoT devices (Veitas and Delaere, 2018a). Contrary to the relatively static and known-in-advance internal network, the external network will be very dynamic and constantly changing, simply due to the movement of the car. In other words, the control system of a connected car will be a perpetually changing fluid assemblage of interacting units and nodes, constantly making decisions about which data and nodes to include in its trusted network and which information to take into account when making driving decisions. The presence of a "core" internal network in such a car does not undermine the fact that the system, in terms of the decision-making capacity, is a an open machine in the Simondon (1980) sense and an extended synthetic cognitive system, capable of change as influenced by external context and information. The intelligence and autonomy of such a car is no longer solely contained in internal algorithms and subsystems, but crucially depends on interaction with the world.



(A) A control system of an *autonomous car* is composed of a network with hundreds of interacting components, where smooth and fast interaction is critical. The types of component range from mission-critical sensors and actuators of a regular car (e.g. on throttle and brake pedals, engine, wheels, windows) to cameras, Lidar units, radars and other sensors needed for autonomous driving. Raw data collected by these nodes is often recorded for legal (e.g. crash investigation) and technical (e.g. perfecting machine learning models) purposes, as well as shared among constituents of the larger ecosystem.



(B) A regular or autonomous car becomes *connected* when embedded into the larger IoT network of external sensors and actuators: mobile devices, road infrastructure, other cars, traffic operators, charging stations, possibly manufacturers or technical support services, all of which can communicate among themselves. Note that even regular modern cars feature integration with mobile phones and downloadable applications, which technically makes them connected.

FIGURE 7.1: Autonomous and connected car networks.

Figure 7.1 provides a very high level overview of autonomous and connected car networks. No less interesting from the perspective of open-ended decentralized computing is the lower level operation of these networks – i.e. how communication of components is actually ensured. A good way to appreciate the paradigm shift is by looking at the historical evolution of vehicular networks. Since components necessarily interact with each other, the first approach was to connect them in peer-to-peer fashion with each other. This is very inefficient as it drives the number of connections to grow exponentially each time a new ECU is added to the network, which happens on a regular basis given the fast development of automotive technology (Tuohy et al., 2015). In the mid 1990s, a bus-based controller area network (CAN) was developed which allows all ECUs to be connected into a physical bus and communicate via passing messages to each other (Van Bulck, Mühlberg, and

Piessens, 2017). Conceptually it therefore can be seen as a kind of message-passing system, albeit still fairly rigid. Speed and correct functioning of the CAN-bus is mission and safety critical for the operation of a car, and therefore it was kept hermetically sealed from infotainment network when the latter was introduced. Yet, with the rising complexity and integration level of infotainment systems, this is no longer true and therefore regular modern cars essentially became *de facto* open systems[6] raising serious security issues (see Section 7.2.2 below). Furthermore, the growing number of ECUs due to the introduction of increasingly advanced levels of autonomous driving and C-ITS capabilities greatly increase the complexity and openness of a vehicular network. A spontaneously established network of several connected cars and road units forms a so-called vehicular ad-hoc network (Mejri, Ben-Othman, and Hamdi, 2014). Clearly, the architecture of vehicular networks has evolved from closed systems with a few known components into open systems with many components, some of which are unknown and spontaneously changing. In the general system theory of Bertalanffy (1968), an open system is defined as a "system in exchange of matter with its environment, presenting import and export, building-up and breaking-down of its material components" (ibid., p. 141). Contrariwise, closed systems are considered to be isolated from their environment. This distinction is analogous to the one of closed and open machines by (Simondon, 1980). The precise computational formulation of the distinction between these two types of systems is proposed on pages 93-94 in terms of the notions of open and closed computing.

Open systems, by the nature of their operation, naturally lend themselves to the models of interactive computation, especially with location and mobility extensions (see Section 5.1.3). The interactivity of decentralized ad-hoc vehicular networks involves considerable uncertainty and nondeterminism in a system with fluid boundaries, which is precisely what open-ended decentralized computing model addresses. From the general computational perspective, an ad-hoc vehicular network of open in-vehicular networks is a flexible scalable structure of computational processes. It is equivalent to an assemblage of lower level assemblages (see Figure 4.6), which is one of the bases of the open-ended decentralized computing model.

We have discussed here autonomous, connected cars, C-ITS and smart mobility systems because they are aspects of the closest to fruition large scale deployment of AI. The advanced implementation phase of these already available technologies provides an array of seemingly "only" technical issues, solutions to which demonstrate the importance of conceptual paradigm shift from closed to open system thinking. Note, however, that most of the concepts discussed above also apply to autonomous advanced robots and AIs in general. A self-driving vehicle is an autonomous robot with a specialized interface for interaction with humans and established infrastructure. Similar principles apply to other advanced systems interacting with humans in a large network – for example healthcare robots – including autonomous and cooperative awareness, connectedness, internal, external sensors and, most importantly, interaction and openness to the world which makes them collectively intelligent.

[6]For example, a simple functionality of the audio system which allows the sound to be automatically increased or decreased depending on the speed of a vehicle breaks the barrier between CAN-bus and infotainment-bus; note additionally the ability to synchronize mobile phones with infotainment systems in order to appreciate the fact that CAN-bus is no longer hermetic (Jan Tobias Mühlberg (2017), private conversation).

7.2.2 Distributed trust, privacy and security

The paradigm shift from closed to open systems thinking in engineering also has a deeply impacting change to how we approach issues of privacy, security and trust management. Closed systems by definition have clearly definable boundaries through which they interact with the world, users and other systems. Traditionally therefore, these boundaries implement security measures for shielding a system from undesirable external influences. Generally, when this shield is broken, or when a malicious actor is able to circumvent it, the whole system is compromised and can no longer be trusted to behave as designed. The picture changes considerably when considering open systems. First of all, open systems do not have clearly defined boundaries and therefore the security shield metaphor is not applicable. Second, open systems can potentially interact with many external actors with initially unknown trust levels. Such systems therefore need to have a dynamic mechanism for selecting relevant and trustworthy interactions from many possibilities. We turn again to the example of a connected car for practical illustration.

As introduced in the previous section, modern regular cars feature mission critical CAN-bus, the security of which is paramount to the safe operation of the car, and an infotainment-bus, which can connect to external inherently insecure systems and networks (e.g. internet). The historically determined logic of looking at a car as a closed system leads to the need for strict separation between CAN-bus (closed subsystem) and infotainment-bus (open subsystem) where any leak of information between the two compromises the security of the whole car. Yet, due to the aspects mentioned above, such separation is available only in theory. It has been repeatedly demonstrated that it is possible to break remotely into the wireless network of a car and assume full control of its mission critical functions by attacking software (i.e. infotainment bus) which is not considered safety-critical but itself interacts with critical components (Miller and Vasalek, 2015). The same has been demonstrated for insulin pumps and general hospital equipment (Noorman et al., 2017) and represents a general problem of smart equipment engineering which spans industries, vendors and models. The "closed system thinking" type response to these issues is to secure the critical components more tightly by putting additional software or physical barriers between closed and open subsystems. Yet, considering even current complexity and desired functionality of in-vehicle networks, such strict compartmentalization is no longer possible. These systems have to be treated as fundamentally open and therefore engineered as such, especially taking into account technological developments towards the connected car, C-ITS and autonomous connected robots in general.

Let us see how the shift in conceptual perspective influences actual engineering decisions. An open system has to establish and ensure secure communications and trust among its vital components, assuming that the network, over which communications happen, itself is untrustworthy. In general computational terms, this is equivalent to the *Byzantine Generals' Problem* of distributed computing (see Box 4.1) where generals (ECUs) communicate via messages in an untrusted environment (CAN-bus) without reliance on the system-level trust provider (the security shield between CAN-bus and infotainment-bus). Note also that, in terms of a trust model, the presence of a system-level trust provider makes a system centralized, while otherwise it is decentralized (see Section 4.3 for the discussion on the "conundrum of decentralization"). Secure communication among decentralized components gives rise to the functionality of a system, which can be represented by a well formed

computation graph (see Figure 4.5). In summary, the in-vehicle network of interacting components (ECUs) corresponds to built-in complete synchrony and consensus in terms of the open-ended decentralized computing model (see page 107) and a structured computation graph.

The security of messages in such a system should be ensured separately for each communication link and communicating component. For that purpose technologies which allow securely encrypted ultra-low latency peer-to-peer communication channels to be established between any ECU in a car are actively developed. These technologies ensure that encryption keys are known only to communicating components and no third party can access or alter the contents (Noorman et al., 2017; Van Bulck, Muhlberg, and Piessens, 2017). Furthermore, each component can carry a unique and unclonable identifier, which is physically derived from its hardware and acts as an encryption key (Herder et al., 2014). The fact that each communication channel can be read and written only by the directly connected nodes themselves, makes the system decentralized in the strict computational sense, where by default there is no global observer which can establish the total order of messages (see Section 4.5.2).

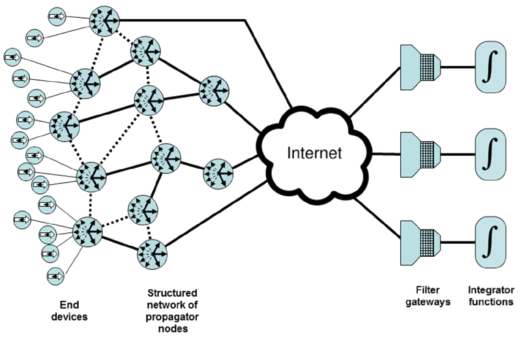
Bottom line, the evolving in-vehicular networks of autonomous cars and ad-hoc vehicular networks of collaborative intelligent transportation systems are examples of physical implementations of fluid assemblages within populations of independent, yet more or less strictly coupled processes following principles of open-ended decentralized computing. This also holds in the larger context of internet of things, where different subsystems can be directly or indirectly organized according to the same principle.

7.2.3 Internet of things and data economy

The term internet of things (IoT) was arguably coined in 1999 in the context of attaching radio frequency identifiers (RFIDs) to physical items for managing the supply chain of a multinational consumer goods corporation. Yet currently it is used to denote, in simple terms, the connectivity of almost all imaginable types of technical objects without direct reference to RFIDs or any other specific technologies (even internet protocols). The inclusion of huge number of different cyber-physical systems into the internet as first class citizens directly grounds the informational network in the physical world (Ashton and others, 2009). Conceptually, IoT is a population of heterogeneous interacting autonomous agents of unprecedented scale and diversity, where data produced and consumed by humans corresponds to only a small share of overall traffic. Examples of networks of such scale and diversity can be found only in natural systems, such as pollen distribution, ant colonies, redwood and rainforest ecosystems (daCosta, 2013). These systems are not and cannot possibly be governed in a centralized manner and are therefore *radically decentralized* by necessity. Computationally, we can model the structure and operation of these networks and their ecologies within the open-ended decentralized computing framework.

Consider for example, the model proposed by daCosta (2013). There, IoT is envisioned as a network of diverse nodes, which can be categorized into four main types by their function in information processing and propagation pipelines (see Figure 7.2a). *End devices* are the smallest and simplest, yet by far the most numerous nodes of the network: temperature, moisture sensors, valve controls, parking meters, home appliances, RFID tags and more. They are almost completely autonomous, having

their own small power source and sending information even if nobody is listening. Furthermore, these are very simple devices: without processor, memory, hard drives and power to accommodate full protocol stack and connect to the internet via their own addresses. Finally, end devices are unreliable by nature, ending operation due to an interrupted power source, a defect or simply because nobody cares to maintain or even monitor them due to their low cost. Yet the power of these devices is in their number and great redundancy[7]. End devices can be likened to pollen – a lot in number, randomly scattered by the wind, but able to "find" and pollinate specific species of plants across vast areas. *Propagator nodes* are more complex devices, responsible for finding, aggregating, annotating, maybe cashing and temporarily storing information from "smart particles". Since end devices use many low-power wireless protocols that are not directly connectable to the internet (see Figure 7.2b for a small sample), propagator nodes house a number of protocol stacks in order to collect this information and make it accessible to the broader network. Importantly, both end devices and propagator nodes will not have stable *a priori* established connections between each other: end nodes will simply beam small information packets at a slow rate, while propagator nodes will find them and propagate them to the internet either directly or through their own ad-hoc mesh networks. *Integrator functions* are a type of more conventional devices or programs that offer advanced analysis, control and human interfaces through a connection to the conventional internet. Furthermore, depending on volumes and velocities of a particular data stream, *filter gateways* may be introduced between the internet and integrator functions in order to clean, reduce, or filter data stream by selected properties.



(A) IoT as a mesh network composed of five components: (1) end devices of low power (mostly sensors), (2) propagator nodes (connecting cheap end devices to the internet) (3) internet (4) filter gateways (selecting relevant) and (5) integrator functions (summarizing data into human-usable information). Note that each node type is a process with inputs and outputs, as defined in Section 4.5.1 on page 111.

Application/ Device	Mode[1]	Chip Size (bytes)	Frequency/ Minute	Repetition %	Effective Data Rate (kbps)	Typical Transmission[2]
Environmental Sensors (Temperature, vibration, strain, moisture, etc.)	S	4.5	1	90%	<1	Z,B,I,W
Lighting Monitoring and Control	B	6.5	1	90%	<1	P,W
Household Appliances	S	4.5	<1	99%	<1	I,P,Z,B
Inventory / Supply Chain	S	4.5	1	95%	<5	R,Z,W
Video Surveillance (Stand By)[3]	B	4.5	60	99%	5	W,Z,C
"Smart" Signs	R	7.5	1800	90%	22	W,Z,P,I
Home Entertainment Control	B	7.5	600	99%	50	I,W
Process Control (Valves, flow rates, etc.)	B	7.5	60	75%	150	Z,W,C
Industrial Machine Control / Diagnosis	B	7.5	6000	25%	340	W,Z,I,C

[1] S = Send-only device; R = Receive-only device; B = Bidirectional
[2] B = Bluetooth; C = Copper (wired); I = Infrared; P = Power line; R = RFID; W = WiFi; Z = ZigBee. Listed in approximate order of suitability/preference.
[3] In stand by mode, local integrator function is processing video surveillance stream, so only "OK" or "fault status is transmitted by IoT interface. When movement of a particular type is detected locally, device would shift to IP mode in order to transit large frames of full video. Local device may also be triggered to full video mode by command from remote integrator function.

(B) A small sub-set of IoT end device types and their diversity as characterized by: (1) data source/sink type, (2) memory size, (3) frequency of signals (4) signal repetition rate, (5) speed of data transmission and (6) physical transmission layer.

FIGURE 7.2: IoT device types and structure. Adapted from daCosta (2013).

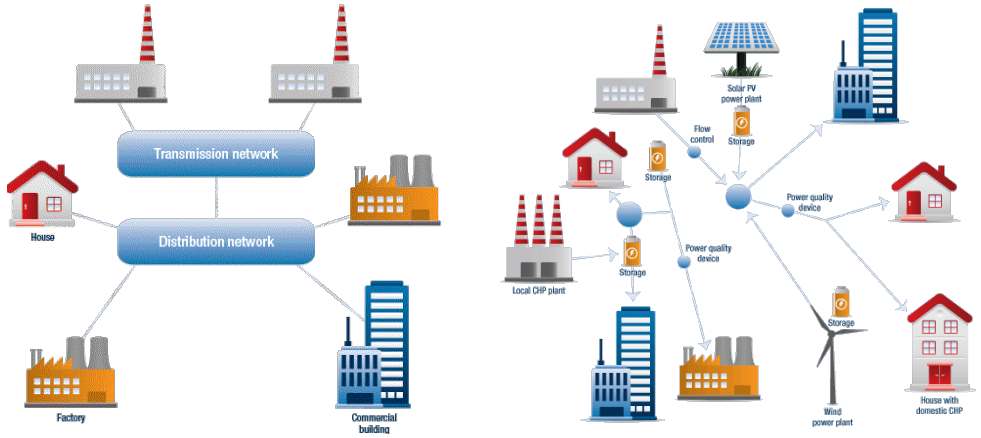
Note that each device of Figure 7.2 is essentially an agent with a process that turns inputs into outputs – an elementary component of the open-ended decentralized computing model (see Section 4.5.1). Information packages – small, larger or

[7] daCosta (2013) gives an example of a bridge which is being monitored by spreading on it thousands of solar-powered vibration sensors. Each sensor is programmed to transmit information once in several minutes or even an hour; furthermore, some sensors will "skip the beat" due to shadow, passing vehicles or other reasons. Yet the aggregated time series data of all sensors can provide adequate information about the state of the bridge at any moment in time.

many aggregated together – are messages passed via the links of diverse types between nodes in the overall network. Importantly, while all devices are parts of a giant network, actual functionalities are due to specific propagation paths of information from end devices to integrator functions and possibly the other way round. These largely self-organized and dynamic paths give rise to multiple communication and, therefore, *computation graphs* – snapshots of immediate configurations of ad-hoc assemblages of network devices. Conceptually, therefore, the internet of things is a vast population of heterogeneous interacting agents and processes which form ad-hoc and fluid assemblages of different functionalities, expressed by their computation graphs. The mechanism of coordination and synchronization in the population can take different forms for different assemblages – e.g. in-built cryptography-based private networks, coupled via blockchain-based smart contracts or guided via the OfferNets type architecture. The economic model will most definitely be an important aspect of the internet of things, where end devices, propagator nodes and integrator functions could belong to different owners, require resources for operation and, therefore, can be seen as agents performing works, which offer outputs and demand inputs in a similar manner as in the offer networks model (see Chapter 6). Collective intelligence of the internet of things will emerge from the functionality of ad-hoc assemblages and interactions between them, rather than being due to the sheer number of nodes, devices and their ubiquitous connectivity.

7.2.4 Energy markets

The smart grid is a decentralized and distributed system from both a technical and governance perspective (Veitas et al., 2015). When fully implemented, it will be an ecosystem of power generators of different sizes and types (traditional large power stations, wind and hydro turbines, solar batteries of power plants and individual homes, fusion devices, etc.). Small power generation capacities will be associated with the new type of market player – *prosumers* – which will both consume and produce energy and trade their daily balances with other prosumers directly or via the global network. Energy balances of most prosumer households will oscillate between the roles of net producers and consumers, depending on usage patterns, actual power generation, and availability of wind or sun. Prosumers may be interconnected into medium-sized local networks – *microgrids* – where they can trade energy balances among themselves. Further, microgrids will be connected to the global energy distribution network, mostly by means of high-voltage lines. They will then offload or demand power from the global network depending on their aggregated balances. High voltage "backbone" networks, while retaining some level of current central control and coordination because of their economy of scale, will be complemented and partially substituted by organically assembling local networks. The smart grid will be equipped with remotely accessible smart meters and synchrophasors (i.e. IoT end devices), collecting data that is available not only for a single central data processing unit or power distribution company, but for the entire network of prosumers. It will allow any entity in the electricity market, be it an economic player or an automated technology, to access and analyse the data from the perspective of its local context. Any electrical appliance, storage or generation unit connected to the smart grid will in principle be able to autonomously negotiate terms of individual electricity supply or demand and in that sense to be an autonomous agent.



(A) Centralized structure of the traditional power grid.

(B) Decentralized structure of the smart grid.

FIGURE 7.3: Transition from traditional power grid to smart grid.

While traditional power grids typically consist of hierarchical structures that make strategic and operational decisions in top-down manner, the smart grid, as a fluid and adaptive system, cannot function in the context of centralized command-and-control. The industry transition from hierarchical governance to the self-organizing nature of decentralized smart grids is hugely challenging technically, managerially and governmentally. Note, however, that from the conceptual and computational perspective, both smart grid and a traditional power grid can be represented in terms of the open-ended decentralized computing model and are quite close to the general offer networks domain model. Over and above interacting heterogeneous agents (power generators, storage units, including electric cars), their assemblages (prosumer households and microgrids) and a stable network structure (low and high voltage lines), smart grids will house an economic exchange network where different participants will demand (input) and offer (output) energy at different prices, times, locations (for example, an electric car can supply and demand energy from its batteries) and possibly other conditions (e.g. sourced from alternative and ecological producers). Such economic networks may or may not use traditional money or barter and nodes will have to engage in direct negotiation before transaction. Furthermore, certain patterns of energy exchange within microgrids or larger networks may stabilize corresponding to a persistent ad-hoc assemblage.

7.2.5 Cloud, edge and fog computing

Notions that are inseparable from the ecosystem of the internet of things are *cloud*, *edge* and *fog* computing. These notions denote layers of infrastructure for powering increasingly complex and interconnected, via the internet, computational resources of global society and economy (see Figure 7.4). Cloud computing refers to the infrastructure that allows a user to hire computing power, database storage, applications and other IT resources physically housed in remote and dedicated data centres. It allows the purchase of computing resources as an on-demand service without the need to care about maintaining or updating the hardware. Fog and edge computing are related and often used interchangeably notions which together mean the push of computing power and machine intelligence closer to where data originates,

i.e. end devices of the internet of things (see Figure 7.2a). However, edge computing also considers devices which are powerful and fully-fledged computers, such as smartphones, tablets, connected cars and more. Cloud, fog and edge computing are different aspects of the same global computing infrastructure and network, and are therefore best approached by looking at their relations rather than by considering them separately.

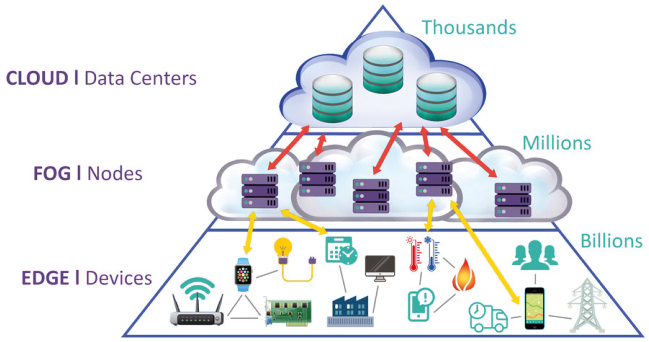


FIGURE 7.4: Relationship between cloud, edge and fog computing in the IoT architecture. Adapted from PubNub (2017).

Consider all computing devices connected to the global network through the internet or otherwise (as in Figure 7.2b). We can see this infrastructure as one giant computer, albeit quite chaotic, decentralized and performing many different computations which are not synchronized with each other. From this perspective, the computing industry's technological trends towards and away from centralized mainframes (1960-1970), distributed client–server (1980-2000), centralized mobile–cloud (2005-2020) and distributed edge intelligence (2020-) (Levine, 2016) are best understood by considering the fundamental principle of decentralized computing (see Section 5.1.2). First, recall that, from the computer-scientific view, any computation and its logics can be represented as a graph of information and data flows between elementary processing units (see Figure 4.5 on page 102). The global computing infrastructure allows the distribution of multiple computation graphs architecturally and physically, depending on the time and memory requirements of each individual processing unit, their assemblages, and the speed and bandwidth of links among them. A computing system assembled within this infrastructure is conceptually equivalent to scalable and structured assemblages of Turing (1948) machines (see Figure 4.6), which furthermore can interact with each other in different ways. Obviously, the fast development of information technologies in terms of reduced processor sizes[8] and data transfer networks' speed and capacity[9] have changed the processing–storing–communication balance when assembling efficient computing graphs.

A far reaching implication, which is partially reflected in cloud, fog and edge computing related trends in information technologies, is the possibility of leveraging location awareness and mobility of individual processes within a computing graph. That is, implementation of computational reflexivity, location awareness and

[8]Modern mobile phones are at least 1000 times faster and have 80 times more memory than state-of-the art mainframe computers of 1970 (Love, 2014).

[9]Fiber optical connections of gigabit speed (1GB/s) start to be routinely offered by internet service providers; fifth generation (5G) cellular service can achieve 10GB/s data rates and sub-millisecond guaranteed latency, which allows to use it in real time mission critical applications (e.g. connected autonomous driving).

mobility, as described with the open-ended decentralized computing model, would allow self-organization and fluidity of computing graphs within IoT with minimal human intervention. It could power synthetic cognitive development processes (see Section 3.3.3) leading to the ecosystem of computing networks with various levels of intelligence in the spirit of the global brain (see Section 2.1.3). Examples of soon-to-be-deployed and current computing ecosystems which use mobility and location awareness functionality are connected cars (see Section 7.2.1) and mobile applications (e.g. mobile games), where processing of the same computation is shared between end devices and servers.

7.2.6 Decentralized applications and computing frameworks

Decentralized applications are software programs which run on architecturally or logically decentralized peer-to-peer computing frameworks and networks. Depending on the business domain and logic of such programs, they are referred to as decentralized organizations (DOs), automated agents (AAs), decentralized autonomous organizations (DAOs) and decentralized autonomous corporations (DACs), all related by the same concept (Raval, 2016). The concept has been lately popularized by blockchain technology and the Ethereum smart contract ecosystem (see Schneider (2014) and Box 4.1). The ideological roots of decentralization of governance are centuries old, while technological parents of modern decentralized applications are the world-wide-web information management network (Berners-Lee, 1989) and early peer-to-peer content distribution networks such as eDonkey2000 (introduced in 2000)[10], Kademlia (2002)[11] or BitTorrent (2001)[12]. Within the scope of this work we take a conceptual and computational perspective to decentralized applications which lets us abstract them from their immediate ideological and technological aspects.

First, note that the concept necessarily involves two aspects: (1) the decentralized applications with specific business logic and (2) the computational framework which executes them. This distinction is important when the framework can handle the execution of not just one but potentially millions of decentralized applications running in parallel. This is precisely the idea of the Ethereum virtual machine (EVM)[13] – the global architecturally decentralized and logically centralized computer. Architecturally, EVM is executed on the decentralized network of independent computing systems and data centres scattered around the world. All nodes of the network run the same software which together implements the Turing (1937) computation model in a parallelized way. Note that the same model can be represented as an emerging global synchrony and consensus within the open-ended decentralized computation model (see Section 4.4.1) owing to the fact that the Turing (1937) model is a special case of actor model (Hewitt, 2006). Another innovation of EVM is the economy of computational processes, where each storage and invocation of a distributed application code costs a certain amount of special-purpose cryptocurrency. Applications can potentially invoke and compensate other applications which are stored on the same network as well as cover their own usage of processing cycles. These functionalities in principle allow the concept of organized networks of decentralized profit seeking autonomous agents in an economic ecosystem.

[10] https://en.wikipedia.org/wiki/EDonkey2000
[11] https://en.wikipedia.org/wiki/Kademlia
[12] https://en.wikipedia.org/wiki/BitTorrent_(software)
[13] Other blockchain based projects of similar nature are EOS, NEO, Cardano and Hyperledger Fabric.

The open-ended decentralized computing model is based on actor model and graph computing, which allows the implementation of decentralized application frameworks in terms of interactive (see page 97) and stigmergic computing (see Section 4.5). It features self-organizing coordination among distributed applications leading to emergence of assemblages of applications with new functionalities. Such self-organization can be supported by a customizable and flexible economic model (see Section 5.1.3). Decentralized application frameworks which explicitly or implicitly follow actor model and are currently in active development include Holochain[14] and SingularityNET[15].

7.3 Summary of the chapter

This chapter presents and discusses six domains of current socio-technological development and their problematic where I deem the perspective informed by the open-ended decentralized computing model and the notion of the "open machine" most valuable. These domains include smart mobility, distributed trust and privacy, internet of things, energy markets and smart grid, cloud, edge and fog computing and, finally, decentralized computing frameworks in general. They by no means cover all possible applications of the computation model, but seem to be most relevant – where the value of the paradigmatic shift that the model proposes is most apparent and readily applicable.

[14]https://holochain.org/

[15]https://singularitynet.io/ (this work was supported by SingularityNET Foundation – see Acknowledgements on page v).

Chapter 8

Summary and conclusion

8.1 Summary

Let us now summarize the path of our design inquiry into the operation of intelligence throughout this work and in the light of the image of "open machine".

Chapter 2 surveys selected concepts, techniques and currents of theoretical and pragmatic thinking in the domain of the evolution of mind, brain and body. It concludes that a cognitive architecture, open in the Simondonian sense, has to: (1) allow identities of cognitive agents to be assembled from a population of lower level interacting elements, (2) account for the ability of cognitive agents to modulate their own sensitivity to environmental interaction, (3) perpetually balance between ordered and disordered states depending on the requirements of this interaction, (4) support the emergence of functional and structural hierarchies, which act (5) as stabilization of patterns of communication among internal and external elements (6) via the mechanism of progressive determination.

Chapter 3 introduces and discusses the open-ended intelligence framework and the philosophy of individuation which integrates the still somewhat isolated principles and aforementioned requirements into the conceptual scheme of synthetic cognitive development. The process of individuation of intelligence happens within a population of interacting primitive elements or agents when patterns of interactions among them stabilize and become more or less persistent. The emergence of stable patterns is driven by signification of meaning of signals and selection of mutually meaningful communications between elements of the population, where meaning is intrinsic to communicating parties only. These patterns give rise to assemblages of different levels of consolidation that correspond to fluid individuals. Furthermore, fluid individuals – partially persistent assemblages of elements – are characterized not only by structured interactions inside their assemblages, but also by structured interactions, across their borders, with other members of the population. Such individuals, embodied or purely computational, are precisely the open machines referred to by Simondon (1980) above. The external elements with which an assemblage interacts across its boundaries constitute together the environment and milieu of a fluid individual.

Assemblages of elements – i.e. fluid individuals – are determined via a history-

dependent evolutionary developmental process where interactivity of elements creates patterns that influence their further interactivity and in this way give rise to the mechanism of progressive determination. This mechanism describes an abstract yet fundamental process of the individuation of cognition, where function determines structure and structure determines function in a progressive way. The process of progressive determination is not directional and can lead to both integration and disintegration of patterns of communication and therefore different levels of fluidity of individuals-assemblages. The scheme of synthetic cognitive development depicts the generalized cognitive development of a cognitive system in terms of sequences of integration and disintegration of internal and external patterns of communication. The very openness of fluid individuals to affect and be affected and evolve due to the interaction via their boundaries implies intelligence.

Chapter 4 applies the computational metaphor to the scheme of synthetic cognitive development and open-ended intelligence in order to construct an implementable model of open-ended decentralized computing. Concretely, we distinguish two notions of computing: closed computing and open computing, where the former is associated with the classical Turing (1937) computing model and the latter with the model of interactive computation and the expanded Turing (1948) model. Further, we position these models in terms of broader notions of centralization and decentralization by utilizing the language of network science and the observation that any computation can be represented in terms of a graph of elementary processes, linked by relations of information and control flows. This leads to the important result that a computation graph can represent and, moreover, itself be seen as a fluid individual constituted of a population of interacting elements that is able to embody the process of progressive determination – by giving rise to different computations as different graph structures. The core of open-ended decentralized computing is precisely this: it is a mechanism for allowing self-organizing integration and disintegration of observable computation graphs of different structures out of a population of heterogeneous interacting elementary processes. Importantly, these structures can be of variable levels of consolidation and determination, and therefore open *and* closed to different degrees.

The framework of open-ended decentralized computing not only allows the representation of different configurations of computation graphs but, moreover, their fluid movement and morphing from one configuration to another. We can understand such a framework as a meta-stable landscape representing different configurations of computation graphs. The system, implemented on the basis of the open-ended decentralized computing model, is meant to be able to reach different configurations via processes of guided or unguided self-organization and interaction with its environment rather than following a pre-defined algorithm. In order to formally describe such a computation, we utilize the concept of stigmergic computing – a computational model of generalized stigmergic cooperation. This model features two components: (1) a computation graph with a fluid structure which is composed of relations among (2) decentralized, autonomous and heterogeneous computation processes. The graph acts as a shared stigmergic medium that enables indirect coordination among processes. Processes leave traces of their interactions on the computation graph which are accessed by other processes that can adjust their behaviours accordingly. Therefore, these processes determine the structure of the fluid assemblage of elementary units of computation. The structure in turn determines further processes and, in this way, enables progressive determination – the mechanism of individuation and open-ended intelligence.

Chapter 5 specifies the semantics of open-ended decentralized computing model in concrete computer-scientific terms, by combining the actor model with graph computing. The actor model enables (1) implementation of a population of heterogeneous computational processes and (2) direct interaction among them via message passing. Graph computing enables (1) the indirect interaction of processes via the shared medium, (2) evolving neighbourhood structure and (3) radical decentralization via vertex-centric traversals. Combining these two computation models allow us to express all the requirements of the mechanism of progressive determination in a single computational framework:

- The actor model is a mathematical model of concurrent computation based on a single computational primitive – a communicating actor. In the open-ended decentralized computing model actors realize elementary components of a population. Such actors can communicate among themselves via passing messages to each other.
- When embedded into a graph structure via a graph computing engine, messages that are passed between actors can be retained as explicit relationships between actor–vertices, effectively creating traces of prior communication. Adhering to the principles of radical decentralization, actor-vertices are able to pass messages only though their immediate neighbours. Graph traversal languages, used for message passing, can nevertheless allow path algebra expressions to be defined, through which the neighbourhood structure of an actor-vertex can be leveraged to reach any other actor in the system. Furthermore, such messages–traversals are sensitive to the graph structure which is shaped by the traces of previous messages.

The semantics of the open-ended decentralized computing model allows the realization of progressive determination and stigmergic computing at a general level, while actual behaviours and desired dynamics of systems implemented on the basis of the model depend on domain specific models – as provided by the example in Chapter 6.

Chapter 6 uses the domain model of a decentralized economic exchange and the problem of search and matching of multi-dimensionally valued goods for implementing open-ended decentralised computing in terms of an actual working code and integration of various existing software frameworks. The implementation amounts to (1) the general simulation modelling engine, (2) monitoring and analysis engine and (3) selected aspects of custom business logic of decentralized exchange. We also performed a number of computational experiments of decentralized search and match for demonstrating the elementary operation of stigmergic computing and the mechanism of progressive determination. In these experiments, a number of different independent concurrent processes interact indirectly via a shared medium – the graph of which they themselves are a part. Importantly, they point to the fundamental trade-off inherent in decentralized computing – that operating in uncertain environments and domains is by definition more computationally costly than optimizing the efficiency of reaching *a priori* defined goals. This corollary has broad implications for the design of generally intelligent AI systems able to address novel situations and problems and come up with creative behaviours – i.e. systems capable to embrace uncertainty. Furthermore, from an even more general perspective, the trade-off can inform and direct design and engineering strategies with proper consideration of desired degree of openness and closeness of any target system.

In a nutshell, this thesis develops the computational model for an open machine, which holds an element of indeterminacy allowing it to be sensitive to uncertain- ties and irregularities which arise from interactions with other humans, artificial intelligences and technical objects within its milieu. The intelligence of interacting open machines, just like biological intelligences, is not defined individually, but rather in reference to ecologies and societies in which they live. With respect to the domain of social science, I am of the opinion that the shift in perspective towards considering a divergent nature of intelligence, cognition and social development, as well as its deeply participatory and decentralized nature, is of the greatest relevance in the current acceleratingly developing hyperconnected world.

8.2 Conclusion

Claude Shannon, mostly known for starting the field of information theory is also considered the pioneer of computer chess as it is known today (Levy, 1988). The ability to play chess skilfully was considered the core of intelligence from the beginning of the modern quest for creating intelligent machines (Newell, Shaw, and Simon, 1958) until a special purpose computer defeated the grandmaster Gary Kasparov, considered the greatest chess player of all times, in 1997. Shannon, however, predicted the conceptual issue with which society, faced with the advances of artificial intelligence, is battling now: "chess is generally considered to require 'thinking' for skilful play; a solution of this problem [by a machine] will force us either to admit the possibility of a mechanized thinking or to [..] restrict our concept of 'thinking'" (Shannon, 1950). Indeed, the reactions to the historical achievements of artificial intelligence have consolidated into a "rule of thumb": every time AI proves itself to be better than the best human in a particular field[1], its abilities are afterwards considered not "real intelligence", despite the appreciation of the same milestone beforehand. Yet, with AI surpassing human capabilities in many tasks and goals, the somewhat hypocritical game of both denying the possibility of synthetic intelligence and avoiding revision of the concept of human "thinking", seems to be reaching a bottleneck due to the game becoming increasingly difficult to play. This bottleneck can turn into a path and opportunity for intelligence expansion via the mediation of technology (Veitas and Weinbaum, 2015) or a binary dilemma of "humans versus machines". In this sense, the quest for creating a synthetic intelligence is not a technical problem, but related to the embodiment of *images of humanity and intelligence* in humanity's collective psyche. They have a self-reinforcing reflective power: "images of humankind which are dominant in a culture are of fundamental importance because they underlie the ways in which the society shapes its institutions, educates its young, and goes about whatever it perceives its business to be" (Markley and Harman, 1981) including, we may add, the creation of its technical objects – as closed-ended instruments of destruction[2] or open-ended companions in evolution[3].

[1] This has happened with board games of immense complexity and social skills (chess, go, poker) as well as a multi-player question answering game played in natural English (Jeopardy!).

[2] See Wikipedia article on lethal autonomous weapons.

[3] Hanson, D. (2017, October). Super-smart robots who care. Presentation presented at the "The future of everything – Sustainable Development in the Age of Rapid Technological Change" Joint Meeting of the United Nations General Assembly Second Committee and ECOSOC, ECOSOC Chamber, UN Headquarters, New York.

Shannon's contemporaries and the minds behind the inception of the modern quest for artificial intelligence had already hinted at the importance of developing two-way communication between computer and human at the end of the 1950s:

> Thus it seems that the rise of effective communication between man and computer will coincide with the rise in the intelligence of the computer – so that the human can say more while thinking less. But [... until we cannot] train them by the blind procedures that work with humans we are caught at the wrong equilibrium of a bistable system: we could design more intelligent machines if we could communicate to them better; we could communicate to them better if they were more intelligent. Limited both in our capabilities for design and communication, every advance in either separately requires a momentous effort. Each success, however, allows a corresponding effort on the other side to reach a little further. At some point the reaction will 'go', and we will find ourselves at the favorable equilibrium point of the system, possessing mechanisms that are both highly intelligent and communicative (Newell, Shaw, and Simon, 1958).

Interactivity and coordination – of particles, elementary processes, neurons, eusocial insects, humans, machines, their assemblages and societies – is the source, catalyst and outcome of the process of becoming intelligent and the fundamental tenet of open-ended intelligence philosophy. In this sense, the concept of individual intelligence and identity is a pragmatically useful narrative: all intelligence is necessarily decentralized and all cognition extended. Following Simondon's treatment of the mode of existence of technical objects[4] quoted on page 176, we repeatedly assert in this work that intelligence is defined by openness and interaction rather than merely by a list, no matter how long and diverse, of skills, properties, abilities or goals. Intelligence reveals itself at the intersection of uncertainty and the ability to make sense of it, just like interesting complex systems reveal themselves at the intersection of chaos and order. When the uncertainty is gone, we no longer see intelligence, only a mechanical manipulation.

I believe that all AI research perspectives could greatly benefit from this approach; but particularly the programme of artificial general intelligence and the ambition to integrate advanced autonomous technology into culture and society in a benevolent way. In this there is no undermining of the importance of skills, properties, abilities and goals or the programme and achievements of narrow artificial intelligence. On the contrary, skills, abilities and goals are products of the process of individuation of intelligence, instrumental for driving it further. However, the open-ended intelligence philosophy and the concept of synthetic cognitive development invites us to see these products not as final results but only temporary stable phases within the broader process, which is driven by self-organizing dynamics. Precisely this approach, if realized through engineering, makes artificial systems – embodying and enabling different skills and goals – open, able to interact and communicate, and therefore truly "autonomously" intelligent. The boundaries between the concept and its implementation are not strict; it is a mistake to separate the conceptual thinking from the engineering and actuation since they are all mutually reinforcing aspects of the same design inquiry.

[4] "Simondon was not merely a philosopher of technology but rather one whose ambition was nothing less than to rewrite the history of philosophy according to the concept of individuation and to invent a philosophical thinking that could effectively integrate technology into culture" (Barthélémy, 2015).

We can relate natural, artificial systems and technical objects with different levels of openness and autonomy by considering the level of their freedom and constraint. The principle of *freedom and constraint* reflects the quest for understanding and designing intelligent systems as a "search" for a balance between (1) openness for change and (2) specific properties in terms of concrete manifestations of intelligent behaviour with respect to a specific context. This is something that evolution – the process of producing cognitive systems, their forms and embodiments – does by progressively determining sets of constraints that define these systems. Yet while determining constraints on cognitive systems and their embodiments, evolution never closes them completely and always leaves a window for change, adaptation and further evolution. Importantly, evolution does not start from concrete properties, skills, goals or survival principles (Weinbaum, 2018, p. 4), which are merely persistent "side effects" of an open-ended process of individuation. The quest for conceiving, designing and engineering intelligent systems, however, unambiguously follows the opposite approach by starting with constraints and relaxing them only when they do not produce "intelligent enough" behaviour. This has worked exceptionally well for the design and engineering of control systems, but starts to shoot the AI programme in the foot by not allowing it to come closer to the very source of intelligence – openness, interactivity and participatory sense-making.

The open-ended decentralized computing model proposes a path for conceiving, designing, simulating and engineering open systems by emphasizing the process of self-organization in the open-ended space of possibilities, rather than explicitly and fully defining its goals and constraints. The model is very general: it itself does not consider any specific constraints, which, obviously, depend on the domain and the environment to which the system will be exposed. A certain number of initial constraints is necessary for bootstrapping the development of an open system able to shape itself further through interaction with its peers. Just as natural organisms need a DNA code and, to a certain level, a benign initial environment for growth, the challenge of open system engineering is to devise these initial conditions in order to bootstrap the mechanism of progressive determination and developmental evolution. Yet, as so much recent research in various fields has demonstrated, communication, interactivity and exposure to uncertainty from the very beginning are needed for a living cognitive system to integrate into its milieu and become intelligent *within it and in relation to it*.

I leave with an excerpt from the Diamond Sutra, the sacred text of Mahayana Buddhism and the world's earliest complete and dated printed book, carrying the timeless message about the illusory and yet tangible nature of individual intelligence, existing only within its milieu as a fleeting instance of a process of becoming:

> Suppose, Subhuti, there were a man endowed with a body, a huge body, so that he had a personal existence like Sumeru, king of mountains. Would that, Subhuti, be a huge personal existence? Subhuti replied: Yes, huge, O Lord [Buddha], huge, O Well-Gone, would his personal existence be. And why so? 'Personal existence, personal existence', as no-existence has that been taught by the Tathagata; for not, O Lord, is that existence or non-existence. Therefore is it called 'personal existence'.
>
> (Diamond Sutra, translated from Sanskrit by Edward Conze).

Bibliography

Abdul-Rahman, Alfarez and Stephen Hailes (1997). "A Distributed Trust Model". In: *Proceedings of the 1997 Workshop on New Security Paradigms*. NSPW '97. Langdale, Cumbria, United Kingdom: ACM, pp. 48–60. ISBN: 0-89791-986-6 (cit. on p. 164).

Abraham, Ajith, Ramos Vitorino, and Crina Grosnan (2006). *Stigmergic Optimization*. 1st ed. Studies in Computational Intelligence 31. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-34689-0 3-540-34689-9 (cit. on pp. 48, 116).

Agha, Gul and Nadeem Jamali (1999). "Concurrent Programming for Distributed Artificial Intelligence". In: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Ed. by Gerhard Weiss. MIT Press, pp. 505–534 (cit. on pp. 122–124, 127, 128, 130).

Agha, Gul A. (Dec. 1986). *Actors: A Model of Concurrent Computation in Distributed Systems*. en. Mit Press. ISBN: 978-0-262-51141-4 (cit. on pp. 105, 122, 125).

Ahmed, E., A. S. Elgazzar, and A. S. Hegazi (June 2005). "An Overview of Complex Adaptive Systems". In: *eprint arXiv:nlin/0506059* (cit. on p. 38).

Albert, Jeong, and Barabasi (July 2000). "Error and attack tolerance of complex networks". In: *Nature* 406.6794, pp. 378–382. ISSN: 1476-4687 (cit. on pp. 140, 141).

Allamanis, Miltiadis, Marc Brockschmidt, and Mahmoud Khademi (2018). "Learning to Represent Programs with Graphs". In: *International Conference on Learning Representations* (cit. on p. 101).

Anadiotis, George (Jan. 2018). *From graph to the world: pioneering a database virtual machine*. English (cit. on p. 133).

Antonopoulos, Andreas M. (2015). *Consensus Algorithms, Blockchain Technology and Bitcoin*. English. Lecture. University College London (cit. on p. 108).

Ashton, Kevin and others (2009). "That 'internet of things' thing". In: *RFID journal* 22.7, pp. 97–114 (cit. on p. 182).

Assunção, Marcos Dias de, Alexandre da Silva Veith, and Rajkumar Buyya (2018). "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions". In: *Journal of Network and Computer Applications* 103, pp. 1–17. ISSN: 1084-8045 (cit. on p. 136).

Barabasi, Albert-Laszlo (2013). *Network Science Book* (cit. on pp. 40, 141).

Barabasi, Albert-Laszlo and Jennifer Frangos (2002). *Linked: The New Science of Networks*. 1st. Perseus Books Group. ISBN: 0-7382-0667-9 978-0-7382-0667-7 (cit. on pp. 41, 140).

Baran, P. (1962). *On Distributed Communications Networks*. English. Technical report P-2626. Santa Monica: RAND Corporation (cit. on p. 99).

— (1964). *On Distributed Communications: I. Introduction to distributed communications networks*. English. Memorandum RM-3420-PR. Santa Monica: RAND Corporation (cit. on p. 99).

Barandiaran, Xabier (2003). "Complexity and evolutionary simulation models in cognitive science Epistemology of synthetic bottom-up simulation modelling". In: (cit. on p. 117).

Baronchelli, Andrea et al. (Apr. 2013). *Networks in Cognitive Science*. arXiv e-print 1304.6736. Trends in Cognitive Science 17(7), 348-360 (2013) (cit. on p. 49).

Barrasa, Jesús (Oct. 2016). *RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?* (Cit. on p. 131).

Barthélémy, Jean-Hugues (Nov. 2015). *Life and Technology: An Inquiry Into and Beyond Simondon*. English. Trans. by Barnaby Norman. Lüneburg: meson press eG. ISBN: 978-3-95796-070-2 (cit. on pp. 38, 193).

Battaglia, P. W. et al. (June 2018). "Relational inductive biases, deep learning, and graph networks". In: *ArXiv e-prints* (cit. on p. 12).

Bennett, Charles H (1995). "Logical depth and physical complexity". In: *The Universal Turing Machine A Half-Century Survey*, pp. 207–235 (cit. on p. 33).

Berger, Toby (1971). "Rate distortion theory: A mathematical basis for data compression". In: (cit. on p. 50).

Berners-Lee, Tim (2007). "Giant global graph". In: *Decentralized Information Group*, p. 29 (cit. on p. 131).

Berners-Lee, Tim, Roy Fielding, and Larry Masinter (Jan. 2005). *Uniform Resource Identifier (URI): Generic Syntax*. Internet standard STD 66 / RFC 3986. Internet Engineering Task Force (cit. on p. 128).

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001). "The semantic web". In: *Scientific american* 284.5, pp. 34–43 (cit. on p. 131).

Berners-Lee, Timothy J (1989). *Information management: A proposal*. Tech. rep. (cit. on p. 187).

Bertalanffy, Ludwig Von (1968). *General System Theory*. Braziller. ISBN: 978-0-8076-0453-3 (cit. on p. 180).

Blockeel, Hendrik, Tijn Witsenburg, and Joost Kok (2007). "Graphs, hypergraphs and inductive logic programming". In: *Proceedings of the 5th International Workshop on Mining and Learning with Graphs*, pp. 93–96 (cit. on p. 157).

Bonabeau, Eric, Marco Dorigo, and Guy Theraulaz (1999). *Swarm Intelligence: From Natural to Artificial Systems*. 1st ed. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, USA. ISBN: 978-0-19-513158-1 0-19-513158-4 (cit. on p. 116).

Bostrom, Nick (June 2012). "The Superintelligent Will: Motivation and Instrumental Rationality in Advanced Artificial Agents". en. In: *Minds and Machines* 22.2, pp. 71–85. ISSN: 0924-6495, 1572-8641 (cit. on p. 4).

Bostrom, Nick and Napoleon Ryan (May 2015). *Superintelligence: Paths, Dangers, Strategies*. English. MP3 Una edition. Audible Studios on Brilliance Audio. ISBN: 978-1-5012-2774-5 (cit. on p. 62).

Brender, Noah Moss (2013). "Sense-Making and Symmetry-Breaking: Merleau-Ponty, Cognitive Science, and Dynamic Systems Theory". In: *Symposium: Canadian Journal of Continental Philosophy/Revue canadienne de philosophie continentale* 17.2, pp. 247–273 (cit. on pp. 39, 55).

Busseniers, Evo (Mar. 2018). "Self-organization versus hierarchical organization: a mathematical investigation of the anarchist philosophy of social organization". English. PhD Thesis. Brussels: Vrije Universiteit Brussel (cit. on p. 47).

Buterin, Vitalik (Feb. 2017). *The Meaning of Decentralization* (cit. on pp. 99, 101).

Callsen, Christian J. and Gul Agha (1994). "Open Heterogeneous Computing in Actor Space". In: *J. Parallel Distrib. Comput* 21.3, pp. 289–300 (cit. on p. 123).

Campbell, Donald (1974). "Evolutionary Epistemology". English. In: *The Philosophy of Karl Popper, Part 1*. Ed. by Paul Arthur Schilpp. La Salle, Ill: Open Court Publishing Co ,U.S., pp. 413–459. ISBN: 978-0-87548-141-8 (cit. on pp. 17, 25).

— (1997). "From Evolutionary Epistemology Via Selection Theory to a Sociology of Scientific Validity". In: *Evolution and Cognition* 3.1-2. Ed. by Cecilia Heyes and Barbara Frankel (cit. on pp. 17, 25, 36).

Campbell, Joseph (1958). *The Symbol Without Meaning*. en. Google-Books-ID: AXTzMgEACAAJ. Rhein-Verlag (cit. on p. 89).

Carhart-Harris, Robin Lester et al. (2014). "The entropic brain: a theory of conscious states informed by neuroimaging research with psychedelic drugs". In: *Frontiers in Human Neuroscience* 8, p. 20 (cit. on pp. 83, 85).

Carnap, Rudolf (1955). "Statistical and Inductive Probability". In: (cit. on p. 90).

Cath, Corinne (2018). "Governing artificial intelligence: ethical, legal and technical opportunities and challenges". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376.2133, p. 20180080 (cit. on p. 176).

Chabot, Pascal, Aliza Krefetz, and Graeme Kirkpatrick (2013). *The Philosophy of Simondon: Between technology and individuation*. 0th ed. Bloomsbury Academic. ISBN: 1-78093-032-1 978-1-78093-032-9 (cit. on pp. 63, 66).

Chan, Serena (2001). "Complex adaptive systems". In: *ESD. 83 Research Seminar in Engineering Systems* (cit. on p. 38).

Chomsky, Noam (1969). *Aspects of the Theory of Syntax*. ISBN: 0-262-53007-4 978-0-262-53007-1 (cit. on p. 12).

Church, Alonzo (1936a). "A note on the Entscheidungsproblem". In: *The journal of symbolic logic* 1.1, pp. 40–41 (cit. on p. 92).

— (1936b). "An unsolvable problem of elementary number theory". In: *American journal of mathematics* 58.2, pp. 345–363 (cit. on pp. 92, 95).

Clark, Andy (2012). "Whatever next? Predictive brains, situated agents, and the future of cognitive science". In: *Behav. Brain Sci* (cit. on p. 55).

Clark, Andy and David J. Chalmers (1998). "The Extended Mind". In: *Analysis* 58.1, pp. 7–19 (cit. on p. 37).

Clinger, William D (1981). *Foundations of Actor Semantics*. English. Technical report. MIT Artificial Intelligence Laboratory (cit. on pp. 122, 123).

Colorni, Alberto, Marco Dorigo, and Vittorio Maniezzo (1991). "Distributed optimization by ant colonies". In: *Proceedings of ECAL91*. Paris: Elsevier Publishing, pp. 134–142 (cit. on p. 116).

Combes, Muriel (2013). *Gilbert Simondon and the Philosophy of the Transindividual*. en. MIT Press. ISBN: 978-0-262-01818-0 (cit. on p. 65).

Cooper, Joel (Mar. 2007). *Cognitive Dissonance: 50 Years of a Classic Theory*. en. SAGE. ISBN: 978-1-84920-344-9 (cit. on p. 84).

Copeland, B Jack and Diane Proudfoot (1999). "Alan Turing's forgotten ideas in computer science". In: *Scientific American* 280.4, pp. 98–103 (cit. on p. 92).

Courtois, P. J. and Robert L. Ashenhurst (1977). *Decomposability. Queueing and Computer System Applications*. First Edition. ACM monograph series. Elsevier Inc, Academic Press Inc. ISBN: 978-0-12-193750-8 0-12-193750-X (cit. on p. 22).

Crumley, Carole L. (1995). "Heterarchy and the Analysis of Complex Societies". en. In: *Archeological Papers of the American Anthropological Association* 6.1, pp. 1–5. ISSN: 1551-8248 (cit. on p. 31).

Cushing, R. S. (2015). *Data-centric computing on distributed resources*. ISBN: 978-94-6259-853-9 (cit. on p. 160).

Cushing, Reginald et al. (2015). "Towards a data processing plane: An automata-based distributed dynamic data processing model". In: *Future Generation Computer Systems*. ISSN: 0167-739X (cit. on p. 160).

Cyganiak, Richard et al. (Feb. 2014). *RDF 1.1 Concepts and Abstract Syntax*. Recommendation. W3C (cit. on p. 131).

daCosta, Francis (Dec. 2013). *Rethinking the Internet of Things: A Scalable Approach to Connecting Everything*. English. 1 edition. Apress (cit. on pp. 182, 183).

Damasio, Antonio (Sept. 2008). *Descartes' Error: Emotion, Reason and the Human Brain*. en. Random House. ISBN: 978-1-4070-7206-7 (cit. on pp. 83, 84).

Dao, Ngoc Ha (2011). "Essays in search and matching theory". PhD Thesis. Université du Québec à Montréal (cit. on p. 150).

Davis, Martin (2004). "The Myth of Hypercomputation". In: *Alan Turing: Life and Legacy of a Great Thinker*. Ed. by Christof Teuscher. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 195–211. ISBN: 978-3-662-05642-4 (cit. on p. 98).

De Jaegher, Hanne and Ezequiel Di Paolo (2007). "Participatory sense-making". In: *Phenomenology and the cognitive sciences* 6.4, pp. 485–507 (cit. on pp. 54, 55).

Dean, Jeffrey and Sanjay Ghemawat (2004). "MapReduce: Simplified Data Processing on Large Clusters". In: *OSDI'04: Sixth Symposium on Operating System Design and Implementation*. San Francisco, CA, pp. 137–150 (cit. on p. 109).

DeLanda, Manuel (Sept. 2006). *A New Philosophy of Society: Assemblage Theory and Social Complexity*. en. London & New York: A&C Black. ISBN: 978-1-4411-1448-8 (cit. on pp. 65–67).

Deleuze, Gilles and Félix Guattari (1983). *Anti-Oedipus: Capitalism and Schizophrenia*. Univ Of Minnesota Press. ISBN: 0-8166-1225-0 978-0-8166-1225-3 (cit. on p. 67).

— (1987). *A Thousand Plateaus: Capitalism and Schizophrenia*. en. Minneapolis: University of Minnesota Press. ISBN: 978-0-8166-1402-8 (cit. on pp. 65, 67).

Di Paolo, Ezequiel A. Di, Marieke Rohde, and Hanneke De Jaegher (2008). "Horizons for the enactive mind: Values, social interaction, and play". en. In: *Enaction: Toward a New Paradigm for Cognitive Science*. Ed. by John Robert Stewart, Olivier Gapenne, and Ezequiel A. Di Di Paolo. MIT Press. ISBN: 978-0-262-01460-1 (cit. on pp. 40, 117).

Di Paolo, Ezequiel Alejandro and Hanne De Jaegher (2012). "The interactive brain hypothesis". In: *Frontiers in Human Neuroscience* 6, p. 163 (cit. on pp. 37, 54, 56).

Dilts, Robert B. and Judith A. Delozier (July 2000). *The Encyclopedia of Systemic NLP and NLP New Coding*. en. Vol. 2. N L P University Press. ISBN: 978-0-9701540-0-2 (cit. on p. 25).

Dina Goldin Scott A. Smolka, Peter Wegner (2006). *Interactive Computation: The New Paradigm*. 1st ed. Springer. ISBN: 978-3-540-34666-1 3-540-34666-X (cit. on p. 97).

Dittrich, P., J. Ziegler, and W. Banzhaf (2001). "Artificial Chemistries - A Review". In: *Artificial life* 7.3, pp. 225–275 (cit. on p. 156).

Dorigo, Marco and Thomas Stützle (2004). *Ant colony optimization*. Bradford Books. MIT Press. ISBN: 978-0-262-04219-2 0-262-04219-3 (cit. on p. 116).

Easley, David and Jon Kleinberg (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. New York, NY, USA: Cambridge University Press. ISBN: 0-521-19533-0 978-0-521-19533-1 (cit. on pp. 131, 141, 150).

Eberbach, Eugene (2000). "Expressiveness of $-Calculus: What Matters?" In: *Intelligent Information Systems*. Heidelberg: Physica-Verlag HD, pp. 145–157. ISBN: 978-3-7908-1846-8 (cit. on p. 98).

Eberbach, Eugene, Dina Goldin, and Peter Wegner (2004). "Turing's Ideas and Models of Computation". In: *Alan Turing: Life and Legacy of a Great Thinker*. Ed. by

Christof Teuscher. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 159–194. ISBN: 978-3-662-05642-4 (cit. on pp. 92, 94, 95).

Edelman, Gerald M. (1987). *Neural Darwinism: the theory of neuronal group selection*. en. New York: Basic Books. ISBN: 978-0-465-04934-9 (cit. on p. 81).

Edelman, Gerald M. and Joseph A. Gally (Aug. 2013). "Reentry: a key mechanism for integration of brain function". In: *Frontiers in Integrative Neuroscience* 7. ISSN: 1662-5145 (cit. on p. 81).

Edelman, Gerald M and Vernon B Mountcastle (1982). *The mindful brain: cortical organization and the group-selective theory of higher brain function*. English. Neurosciences Research Program. Cambridge: MIT Press. ISBN: 0-262-55007-5 978-0-262-55007-9 0-262-05020-X 978-0-262-05020-3 (cit. on p. 55).

Edelman, Gerald M. and Giulio Tononi (2000). *A Universe of Consciousness: How Matter Becomes Imagination*. en. Basic Books. ISBN: 0-465-01377-5 (cit. on pp. 24, 70, 80, 83).

Eldredge, Niles (Sept. 2016). "The Checkered Career of Hierarchical Thinking in Evolutionary Biology". English. In: *Evolutionary Theory: A Hierarchical Perspective*. Ed. by Niles Eldredge et al. Reprint edition. Chicago: University Of Chicago Press, pp. 1–17. ISBN: 978-0-226-42622-8 (cit. on p. 16).

Emde Boas, Peter van (2012). "Turing Machines for Dummies: Why Representations Do Matter". In: *Proceedings of the 38th International Conference on Current Trends in Theory and Practice of Computer Science*. SOFSEM'12. Berlin, Heidelberg: Springer-Verlag, pp. 14–30. ISBN: 978-3-642-27659-0 (cit. on p. 101).

Erb, Benjamin (Apr. 2012). "Concurrent Programming for Scalable Web Architectures". Diploma Thesis. Institute of Distributed Systems, Ulm University (cit. on p. 124).

Erdös, P. and A. Rényi (1959). "On Random Graphs I". In: *Publicationes Mathematicae Debrecen* 6, p. 290 (cit. on p. 168).

European Parliament (Feb. 2017). *Resolution with recommendations to the Commission on Civil Law Rules on Robotics (2015/2103(INL))*. English (cit. on p. 176).

Firn, Richard (2004). "Plant Intelligence: an Alternative Point of View". In: *Annals of Botany* 93.4, pp. 345–351 (cit. on p. 76).

Flynn, Bernard (2011). "Maurice Merleau-Ponty". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2011 (cit. on p. 69).

Fox, Dieter et al. (2001). "Particle filters for mobile robot localization". In: *Sequential Monte Carlo methods in practice*. Springer, pp. 401–428 (cit. on p. 18).

Gell-Mann, Murray (1994). "Complex adaptive systems". In: *Complexity: Metaphors, models and reality*, pp. 17–45 (cit. on p. 38).

Gershenson, Carlos (June 2010). "Computing Networks: A General Framework to Contrast Neural and Swarm Cognitions". In: *Paladyn* 1.2. arXiv: 1001.5244, pp. 147–153. ISSN: 2080-9778, 2081-4836 (cit. on p. 114).

— (2011). "Self-Organization Leads to Supraoptimal Performance in Public Transportation Systems". In: *PLOS ONE* 6.6, pp. 1–6 (cit. on p. 178).

Gigerenzer, Gerd (Jan. 2008). "Why Heuristics Work". en. In: *Perspectives on Psychological Science* 3.1, pp. 20–29. ISSN: 1745-6916 (cit. on p. 50).

Gloag, Erin S., Lynne Turnbull, and Cynthia B. Whitchurch (Jan. 2015). "Bacterial Stigmergy: An Organising Principle of Multicellular Collective Behaviours of Bacteria". en. In: *Scientifica* 2015, e387342 (cit. on p. 48).

Gode, Dhananjay K and Shyam Sunder (1993). "Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality". In: *Journal of political economy* 101.1, pp. 119–137 (cit. on p. 149).

Goertzel, B. (Mar. 2017a). "Toward a Formal Model of Cognitive Synergy". In: *ArXiv e-prints* (cit. on p. 157).

Goertzel, Ben (2002). *Creating Internet Intelligence: Wild Computing, Distributed Digital Consciousness, and the Emerging Global Brain*. en. Google-Books-ID: w5nbBwAAQBAJ. Springer Science & Business Media. ISBN: 978-1-4615-0561-7 (cit. on p. 13).

— (2009). "Toward a Formal Characterization of Real-World General Intelligence". In: *Proceedings of the 3rd Conference on Artificial General Intelligence*. AGI, pp. 19–24 (cit. on p. 12).

— (2014). "Characterizing Human-Like Consciousness: An Integrative Approach". English. In: *Procedia Computer Science* 41, pp. 152–157 (cit. on p. 176).

— (May 2015a). "Beyond Money: Offer Networks, a Potential Infrastructure for a Post-Money Economy". English. In: *The End of the Beginning: Life, Society and Economy on the Brink of the Singularity*. Ed. by Ben Goertzel and Ted Goertzel. 1 edition. Humanity+ Press, pp. 522–554. ISBN: 978-0-692-45766-5 (cit. on pp. 149, 151, 157).

— (2015b). "Matching Algorithm For Attention Offer Networks". English. v3 (cit. on p. 156).

Goertzel, Ben, Cassio Pennachin, and Neil Geisweiller (Feb. 2014). *I. Engineering General Intelligence, Part 1: A Path to Advanced AGI via Embodied Learning and Cognitive Synergy*. English. 2014 edition. New York: Atlantis Press. ISBN: 978-94-6239-026-3 (cit. on pp. 11, 12, 101).

Goertzel, Zar (Aug. 2017b). "Offer Networks Simulation and Dynamics". English. MA thesis. Copenhagen (cit. on pp. 156, 159).

Goldin, Dina and Peter Wegner (2002). "Paraconsistency of Inetractive Computation". English. In: *Proceedings of the International Workshop on Paraconsistent Computational Logic*. Ed. by Henrik Decker, Jørgen Villadsen, and Toshiharu Waragai. Datalogiske tidsskrifter, Roskilde University, Denmark (cit. on p. 98).

— (2006). "Principles of Interactive Computation". In: *Interactive Computation: The New Paradigm*. Ed. by Dina Goldin, Scott A. Smolka, and Peter Wegner. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 25–37. ISBN: 978-3-540-34874-0 (cit. on pp. 95, 98).

Gontier, Nathalie (2006). "Evolutionary Epistemology". In: *Internet Encyclopedia of Philosophy* (cit. on pp. 17, 37).

Gonzalez, Joseph E et al. (2012). "PowerGraph: distributed graph-parallel computation on natural graphs". In: *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*. USENIX Association, pp. 17–30 (cit. on p. 109).

Gouws, Stephen, G-J van Rooyen, and Herman A. Engelbrecht (Oct. 2010). "Measuring Conceptual Similarity by Spreading Activation over Wikipedia's Hyperlink Structure". In: *Proceedings of the {2nd} Workshop on {The People's Web Meets NLP: Collaboratively Constructed Semantic Resources}*. Beijing, China: Coling 2010 Organizing Committee, pp. 46–54 (cit. on p. 49).

Granovetter, Mark (1985). "Economic action and social structure: The problem of embeddedness". In: *American journal of sociology* 91.3, pp. 481–510 (cit. on p. 150).

Grassé, Pierre-Paul (Mar. 1959). "La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs". In: *Insectes Sociaux* 6.1, pp. 41–80. ISSN: 0020-1812 (cit. on p. 47).

Greif, Irene (1975). "Semantics of communicating parallel processes". PhD Thesis. Massachusetts Institute of Technology, Cambridge, MA, USA (cit. on pp. 122, 125).

Griffiths, Paul E and James Tabery (2013). "Developmental systems theory: What does it explain, and how does it explain it". In: *Adv Child Dev Behav* 44, pp. 65–94 (cit. on pp. 37, 38).

Growney, JoAnne Simpson (1982). "Planning for Interruptions". In: *Mathematics Magazine* 55.4, pp. 213–219 (cit. on p. 21).

Grünwald, Peter and Paul Vitányi (2010). "Shannon Information and Kolmogorov Complexity". In: (cit. on p. 33).

Gödel, Kurt (1931). "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I". Trans. by B Meltzer. In: *Monatshefte für Mathematik und Physik* 38.1, pp. 173–198 (cit. on pp. 92, 95, 98).

Haber, Stuart and W. Scott Stornetta (Jan. 1991). "How to time-stamp a digital document". In: *Journal of Cryptology* 3.2, pp. 99–111. ISSN: 1432-1378 (cit. on p. 100).

Hadaller, David, Kevin Regan, and Tyrel Russell (2005). *Necessity of supernodes survey*. Tech. rep. Technical report, Technical Report 2005-1, Department of Computer Science ... (cit. on p. 141).

Hall, Brian K. (1999). "Evolution as the Control of Development by Ecology". en. In: *Evolutionary Developmental Biology*. Springer, Dordrecht, pp. 297–306. ISBN: 978-0-412-78590-0 978-94-011-3961-8 (cit. on p. 35).

Hansson, Sven Ove and Till Grüne-Yanoff (2018). "Preferences". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2018. Metaphysics Research Lab, Stanford University (cit. on p. 156).

Harmon-Jones, Eddie (2012). "Cognitive Dissonance Theory". In: *Encyclopedia of Mind*. Ed. by Harold Pashler et al. Thousand Oaks, CA: Sage Publications (cit. on pp. 83, 84).

Harnad, Stevan (June 1990). "The symbol grounding problem". In: *Physica D: Nonlinear Phenomena* 42.1–3, pp. 335–346. ISSN: 0167-2789 (cit. on p. 88).

Harris, Steve, Andy Seaborne, and Eric Prud'hommeaux (Mar. 2013). *SPARQL 1.1 Query Language*. Recommendation. W3C (cit. on p. 131).

Harrison, Karey (2013). "Building resilient communities". In: *M/C Journal* 16.5 (cit. on p. 71).

Hawkins, Jeff and Sandra Blakeslee (July 2005). *On Intelligence*. English. New York: St. Martin's Griffin. ISBN: 0-8050-7853-3 (cit. on pp. 28, 47).

Heckel, Reiko, Alexander Kurz, and Edmund Chattoe-Brown (2017). "Features of Agent-based Models". In: *arXiv preprint arXiv:1712.09496* (cit. on p. 41).

Herder, C. et al. (Aug. 2014). "Physical Unclonable Functions and Applications: A Tutorial". In: *Proceedings of the IEEE* 102.8, pp. 1126–1141. ISSN: 0018-9219 (cit. on p. 182).

Hewitt, C. (Aug. 2010). "Actor Model of Computation: Scalable Robust Information Systems". In: *ArXiv e-prints* (cit. on pp. 124, 126).

— (2013). "What is Computation? Actor Model vs Turing's Model". en. In: *A Computable Universe: Understanding and Exploring Nature as Computation*. Ed. by Hector Zenil and Roger Penrose. World Scientific, pp. 159–187. ISBN: 978-981-4374-29-3 (cit. on pp. 124, 125).

Hewitt, Carl (Dec. 1976). "Viewing Control Structures as Patterns of Passing Messages". en_US. In: (cit. on pp. 105, 122, 125).

— (2006). "What Is Commitment? Physical, Organizational, and Social". In: *Coordination, Organizations, Institutions, and Norms in Agent Systems II - AAMAS 2006 and ECAI 2006 International Workshops, COIN 2006 Hakodate, Japan, May 9, 2006 Riva del Garda, Italy, August 28, 2006. Revised Selected Papers*, pp. 293–307 (cit. on pp. 126, 187).

Hewitt, Carl (Mar. 2017). "Actor Model of Computation for Scalable Robust Information Systems". In: *Symposium on Logic and Collaboration for Intelligent Applications*, Stanford, United States (cit. on p. 124).

Hewitt, Carl, Peter Bishop, and Richard Steiger (1973). "A Universal Modular ACTOR Formalism for Artificial Intelligence". In: *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*. IJCAI'73. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 235–245 (cit. on pp. 98, 122, 125).

Hewitt, Carl and Carl Manning (1996). "Participatory Semantics for Multi-Agent Systems". English. In: *Proceedings of the second international conference on multiagent systems*. Kyoto: AAAI Press, pp. 441–442. ISBN: ISBN 978-1-57735-013-2 (cit. on p. 125).

Heyes, Cecilia M. and David L. Hull (Aug. 2001). *Selection Theory and Social Construct: The Evolutionary Naturalistic Epistemology of Donald T. Campbell*. en. SUNY Press. ISBN: 978-0-7914-9017-4 (cit. on p. 17).

Heylighen, F. (2008). "Accelerating socio-technological evolution: from ephemeralization and stigmergy to the global brain". In: *Globalization as evolutionary process: modeling global change*. Vol. 10, p. 284 (cit. on p. 49).

Heylighen, Francis (1995). "(Meta)Systems as Constraints on Variation— a Classification and Natural History of Metasystem Transitions". In: *World Futures* 45.1, pp. 59–85 (cit. on pp. 17, 20).

— (July 2001a). "Bootstrapping knowledge representations: From entailment meshes via semantic nets to learning webs". In: *Kybernetes* 30.5/6, pp. 691–725. ISSN: 0368-492X (cit. on p. 49).

— (2001b). *The science of self-organization and adaptivity*. Ed. by L. D. Kiel. Oxford (cit. on p. 39).

— (2002a). "The global brain as a new utopia". In: *Zukunftsfiguren, Suhrkamp, Frankurt* (cit. on pp. 31, 46, 49).

— (2002b). "The Global Superorganism: an evolutionary-cybernetic model of the emerging network society". In: *Journal of Social and Evolutionary Systems* (cit. on p. 12).

— (2009a). *Complexity and Self-organization*. Ed. by M.J. Bates and M.N. Maack (cit. on pp. 39, 40).

— (2009b). "Coognitive Systems: A Cybernetic Perspective On The New Science Of Mind". Brussels (cit. on p. 49).

— (2011). "Self-organization of complex, intelligent systems: an action ontology for transdisciplinary integration". In: *Integral Review* (cit. on pp. 28, 48, 155).

— (Jan. 2013). "Self-organization in Communicating Groups: The Emergence of Coordination, Shared References and Collective Intelligence". en. In: *Complexity Perspectives on Language, Communication and Society*. Ed. by Àngels Massip-Bonet and Albert Bastardas-Boada. Understanding Complex Systems. Springer Berlin Heidelberg, pp. 117–149. ISBN: 978-3-642-32816-9 978-3-642-32817-6 (cit. on pp. 81, 103, 155).

— (2015a). "Complexity and Evolution: fundamental concepts of a new scientific worldview". Brussels (cit. on p. 70).

— (2015b). "Stigmergy as a Universal Coordination Mechanism: components, varieties and applications". In: *To appear in: Human Stigmergy: Theoretical Developments and New Applications. Springer.* (Cit. on pp. 47, 48).

— (June 2016). "Stigmergy as a universal coordination mechanism I: Definition and components". In: *Cognitive Systems Research*. Special Issue of Cognitive Systems Research – Human-Human Stigmergy 38, pp. 4–13. ISSN: 1389-0417 (cit. on pp. 47, 48, 74).

— (Jan. 2017). "The offer network protocol: Mathematical foundations and a roadmap for the development of a global brain". In: *The European Physical Journal Special Topics* 226.2, pp. 283–312. ISSN: 1951-6401 (cit. on pp. 149, 156).

Heylighen, Francis and Johan Bollen (1996). "The World-Wide Web as a Super-Brain: from metaphor to model". In: *Cybernetics and Systems '96. R. Trappl (Ed.). Austrian Society For Cybernetics*. Press, pp. 917–922 (cit. on p. 49).

Heylighen, Francis and Nathalie Gontier (2019). "Transcending the Rational Symbol System: how information and communication technology integrates science, art, philosophy and spirituality into a global brain". In: *Handbook of Human Symbolic Evolution*. Forthcoming. Oxford University Press (cit. on p. 89).

Heylighen, Francis and Cliff Joslyn (1995). "Towards a theory of Metasystem transitions: Introduction to the special issue". In: *World Futures* 45.1-4, pp. 1–4 (cit. on p. 26).

Hoare, C. A. R. (Aug. 1978). "Communicating Sequential Processes". In: *Commun. ACM* 21.8, pp. 666–677. ISSN: 0001-0782 (cit. on p. 98).

Houck, Christopher R. and Gul Agha (1992). "HAL: A High-Level Actor Language and Its Distributed Implementation". In: *Proceedings of the 1992 International Conference on Parallel Processing, University of Michigan, An Arbor, Michigan, USA, August 17-21, 1992. Volume II: Software.* Pp. 158–165 (cit. on p. 123).

Humboldt von, Wilhelm Freiherr (1988). *On Language: The Diversity of Human Language-Structure and its Influence on the Mental Development of Mankind*. Texts in German Philosophy. Cambridge University Press. ISBN: 0-521-31513-1 978-0-521-31513-5 (cit. on p. 12).

Hutter, Marcus (Oct. 2012). *Universal Artificial Intelligence*. Presentation by Marcus Hutter at Singularity Summit Australia 2012. Singularity Summit Australia 2012 (cit. on p. 13).

— (2013). "To Create a Super-Intelligent Machine, Start with an Equation". In: *The Conversation* November.29, pp. 1–5 (cit. on p. 13).

Immerman, Neil (2016). "Computability and Complexity". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2016. Metaphysics Research Lab, Stanford University (cit. on p. 91).

Jackson, William A (2006). "On the social structure of markets". In: *Cambridge Journal of Economics* 31.2, pp. 235–253 (cit. on p. 149).

James, William (1890). *The Principles of Psychology*. English. Vol. 1. American science series. New York: Henry Holt and Company (cit. on p. 15).

Johanson, Jan and Lars-Gunnar Mattsson (1994). "The Markets-As-Networks Tradition in Sweden". In: *Research traditions in marketing*. Ed. by Gilles Laurent, Gary L. Lilien, and Bernard Pras. Dordrecht: Springer Netherlands, pp. 321–346. ISBN: 978-94-011-1402-8 (cit. on p. 150).

Johnson, John and Adrian V Gheorghe (2013). "Antifragility analysis and measurement framework for systems of systems". In: *International Journal of Disaster Risk Science* 4.4, pp. 159–168 (cit. on p. 40).

Jones, Matt et al. (July 2013). "Sequential effects in response time reveal learning mechanisms and event representations". eng. In: *Psychological Review* 120.3, pp. 628–666. ISSN: 1939-1471 (cit. on pp. 33, 34).

Joseph, Shibily, Jasmin E.A., and Soumya Chandran (2015). "Stream Computing: Opportunities and Challenges in Smart Grid". In: *Procedia Technology* 21, pp. 49–53. ISSN: 2212-0173 (cit. on p. 136).

Joslyn, Cliff, Francis Heylighen, and Valentin Turchin (1992). "Metasystem Transition Theory". In: *Principia Cybernetica Web*. Ed. by F. Heylighen, Cliff Joslyn, and Valentin Turchin. Brussels: Principia Cybernetica (cit. on p. 25).

Kahrs, Herman (2017). *Creating Human Level AI: How and When | Ray Kurzweil* (cit. on pp. 12, 176).

Kalinka, Alex T. and Pavel Tomancak (July 2012). "The evolution of early animal embryos: conservation or divergence?" eng. In: *Trends in Ecology & Evolution* 27.7, pp. 385–393. ISSN: 1872-8383 (cit. on p. 35).

Karnow, Curtis EA (1996). "Liability for distributed artificial intelligences". In: *Berkeley Technology Law Journal*, pp. 147–204 (cit. on p. 176).

Kegan, Robert (1982). *The evolving self: problem and process in human development*. English. Cambridge, Mass.: Harvard University Press. ISBN: 0-674-27230-7 978-0-674-27230-9 0-674-27231-5 978-0-674-27231-6 (cit. on pp. 53, 84).

Kelly, Kevin (1998). "The Computational Metaphor". English. In: *The Whole Earth Catalog* Winter (cit. on p. 87).

— (2010). *What Technology Wants*. Viking Adult. ISBN: 978-1-101-44446-7 (cit. on p. 87).

Kivelä, Mikko et al. (2014). "Multilayer networks". In: *Journal of complex networks* 2.3, pp. 203–271 (cit. on p. 41).

Kleedorfer, Florian and Christina Maria Busch (2013). "Beyond Data: Building a Web of Needs". In: *LDOW* (cit. on p. 156).

Klein, G., B. Moon, and R.R. Hoffman (July 2006). "1. Making Sense of Sensemaking 1: Alternative Perspectives". In: *IEEE Intelligent Systems* 21.4, pp. 70–73. ISSN: 1541-1672 (cit. on p. 55).

Kleinberg, Jon M et al. (1999). "The web as a graph: measurements, models, and methods". In: *International Computing and Combinatorics Conference*. Springer, pp. 1–17 (cit. on p. 133).

Kohlberg, Lawrence and Carol Gilligan (1971). "The adolescent as a philosopher: The discovery of the self in a postconventional world". In: *Daedalus*, pp. 1051–1086 (cit. on pp. 53, 54).

Kolokolova, Antonina (2017). "Complexity Barriers as Independence". In: *The Incomputable: Journeys Beyond the Turing Barrier*. Ed. by S. Barry Cooper and Mariya I. Soskova. Cham: Springer International Publishing, pp. 143–168. ISBN: 978-3-319-43669-2 (cit. on p. 91).

Kolonin, A. et al. (June 2018). "A Reputation System for Artificial Societies". In: *ArXiv e-prints* (cit. on pp. 108, 164).

Komosinski, Maciej and Andrew Adamatzky (June 2009). *Artificial Life Models in Software*. en. Springer Science & Business Media. ISBN: 978-1-84882-285-6 (cit. on p. 40).

Kshemkalyani, Ajay D. and Mukesh Singhal (Apr. 2008). *Distributed Computing: Principles, Algorithms, and Systems*. en. Cambridge University Press. ISBN: 978-1-139-47031-5 (cit. on p. 140).

Kurzweil, Ray (2005). *The Singularity is Near: When Humans Transcend Biology*. en. Viking. ISBN: 978-0-670-03384-3 (cit. on p. 12).

Laland, Kevin, Blake Matthews, and Marcus W. Feldman (2016). "An introduction to niche construction theory". In: *Evolutionary Ecology* 30, pp. 191–202. ISSN: 0269-7653 (cit. on p. 36).

Lamport, Leslie, Robert Shostak, and Marshall Pease (1982). "The Byzantine generals problem". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3, pp. 382–401 (cit. on p. 100).

Latour, Bruno (Sept. 2007). *Reassembling the Social: An Introduction to Actor-Network-Theory*. en. OUP Oxford. ISBN: 978-0-19-925605-1 (cit. on pp. 65, 176).

Laud, Peeter (2011). *Complexity Theory (MTAT.07.004), Lecture 2*. Tech. rep. (cit. on p. 102).

Lebel, Catherine et al. (2008). "Microstructural maturation of the human brain from childhood to adulthood". In: *Neuroimage* 40.3, pp. 1044–1055 (cit. on p. 16).

Legg, Shane (2008). "Machine Super Intelligence". Engels. PhD thesis. University of Lugano (cit. on p. 13).

Legg, Shane and Marcus Hutter (2007). "A collection of definitions of intelligence". In: *Advances in artificial general intelligence: Concepts, archtectures and algorithms*. Ed. by Ben Goertzel and Pei Wang. Vol. 157. Frontiers in Artificial Intelligence and applications. Amsterdam: IOS Press, pp. 17–24 (cit. on p. 11).

Leuteritz, Thomas EJ and Hamid R Ekbia (2008). "Not all roads lead to resilience: a complex systems approach to the comparative analysis of tortoises in arid ecosystems". In: *Ecology and Society* 13.1, p. 1 (cit. on p. 71).

Levine, Peter (Nov. 2016). *Return to the Edge and the End of Cloud Computing*. English. Presentation (cit. on p. 186).

Levy, David, ed. (1988). *Computer chess compendium*. 1st ed. New York: Springer-Verlag (cit. on p. 192).

Lewontin, Richard C (1970). "The units of selection". In: *Annual review of ecology and systematics* 1.1, pp. 1–18 (cit. on p. 19).

Li, Hongxia et al. (2016). "Distributed adaptive consensus of heterogeneous multi-agent systems with unknown coupling weights". In: *IMA Journal of Mathematical Control and Information* 35.1, pp. 93–105 (cit. on p. 116).

Li, Ming and Paul Vitanyi (Feb. 1997). *An Introduction to Kolmogorov Complexity and Its Applications*. en. Google-Books-ID: LKEmB_GQ53QC. Springer Science & Business Media. ISBN: 978-0-387-94868-3 (cit. on p. 28).

Li, S. et al. (Jan. 2018). "A Fundamental Tradeoff Between Computation and Communication in Distributed Computing". In: *IEEE Transactions on Information Theory* 64.1, pp. 109–128. ISSN: 0018-9448 (cit. on p. 126).

Lloyd, Elisabeth (2012). "Units and Levels of Selection". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2012 (cit. on p. 18).

Love, Dylan (May 2014). "The Specs On This 1970 IBM Mainframe Will Remind You Just How Far Technology Has Come". English. In: *Business Insider* (cit. on p. 186).

M. Collins, Allan and Elizabeth Loftus (1975). "A Spreading Activation Theory of Semantic Processing". In: *Psychological Review* 82, pp. 407–428 (cit. on p. 49).

MacKay, Donald M. (1969). *Information, mechanism and meaning*. M.I.T. Press. ISBN: 0-262-13055-6 978-0-262-13055-4 0-262-63032-X 978-0-262-63032-0 (cit. on pp. 65–67, 89, 90).

Madison, Michael et al. (2015). "Nosql database technologies". In: *Journal of International Technology and Information Management* 24.1, p. 1 (cit. on p. 131).

MaidSafe (Dec. 2015). *Evolving Terminology with Evolved Technology: Decentralized versus Distributed* (cit. on p. 99).

Mandler, Michael (Feb. 2005). "Incomplete preferences and rational intransitivity of choice". In: *Games and Economic Behavior* 50.2, pp. 255–277. ISSN: 0899-8256 (cit. on pp. 150, 164).

Mansell, Warren and Richard S. Marken (2015). "The origins and future of control theory in psychology". In: *Review of General Psychology* 19.4, pp. 425–430. ISSN: 1939-1552(Electronic),1089-2680(Print) (cit. on pp. 43, 44, 47).

Markley, O. W. and Willis W. Harman, eds. (Sept. 1981). *Changing Images of Man*. Pergamon Press. ISBN: 0-08-024313-4 (cit. on pp. 56, 192).

Marsh, Leslie and Christian Onof (2008). "Stigmergic epistemology, stigmergic cognition". In: *Cognitive Systems Research* 9.1, pp. 136–149 (cit. on pp. 48, 72, 74, 109).

Matt, Christian (Aug. 2018). "Fog Computing". In: *Business & Information Systems Engineering* 60.4, pp. 351–355. ISSN: 1867-0202 (cit. on p. 133).

Maturana, H. R. and Francisco J. Varela (Jan. 1980). *Autopoiesis and Cognition: The Realization of the Living*. en. Dordrecht: D. Reidel Publishing Company. ISBN: 978-90-277-1016-1 (cit. on p. 80).

McCarthy, John et al. (2006). "A proposal for the Dartmouth summer research project on artificial intelligence, August 31, 1955". In: *AI magazine* 27.4, p. 12 (cit. on pp. 8, 11).

McClelland, Kent (Dec. 1994). "Perceptual Control and Social Power". en. In: *Sociological Perspectives* 37.4, pp. 461–496. ISSN: 0731-1214 (cit. on p. 46).

McClelland, Kent A. (2006). "Understanding Collective Control Processes". en. In: *Purpose, Meaning, and Action*. Palgrave Macmillan, New York, pp. 31–56. ISBN: 978-1-349-73419-1 978-1-137-10809-8 (cit. on p. 47).

McCulloch, Warren S. (June 1945). "A heterarchy of values determined by the topology of nervous nets". en. In: *The bulletin of mathematical biophysics* 7.2, pp. 89–93. ISSN: 0007-4985, 1522-9602 (cit. on p. 31).

McCune, Robert Ryan, Tim Weninger, and Greg Madey (Oct. 2015). "Thinking Like a Vertex: A Survey of Vertex-Centric Frameworks for Large-Scale Distributed Graph Processing". In: *ACM Comput. Surv.* 48.2, 25:1–25:39. ISSN: 0360-0300 (cit. on p. 134).

McGann, Marek (2008). *Enactive Cognition: A Cognition Briefing*. English (cit. on p. 55).

Mejri, Mohamed Nidhal, Jalel Ben-Othman, and Mohamed Hamdi (2014). "Survey on VANET security challenges and possible cryptographic solutions". In: *Vehicular Communications* 1.2, pp. 53–66. ISSN: 2214-2096 (cit. on p. 180).

Miller, Chris and Chris Vasalek (Aug. 2015). *Remote exploitationof an unaltered passenger vehicle*. English. Presentation. Las Vegas, NV (cit. on p. 181).

Miller, George A., Eugene Galanter, and Karl H. Pribram (1960). *Plans and the structure of behavior*. en. Hole, Rinehart and Winston, Inc. ISBN: 0-03-010075-5 (cit. on pp. 24, 25, 45).

Milner, Robin (Jan. 1993a). "Elements of Interaction: Turing Award Lecture". In: *Commun. ACM* 36.1, pp. 78–89. ISSN: 0001-0782 (cit. on pp. 95, 97, 98).

— (1993b). "The polyadic -calculus: a tutorial". In: *Logic and algebra of specification*. Springer, pp. 203–246 (cit. on p. 98).

Minati, Gianfranco, Eliano Pessa, and Mario Abram (2009). *Processes of Emergence of Systems and Systemic Properties: Towards a General Theory of Emergence, Proceedings of the International Conference, Castel Ivano,Italy, 18-20 Ocotber 2007*. World Scientific Publishing Company. ISBN: 981-279-346-1 (cit. on p. 31).

Minsky, Marvin (Mar. 1988). *Society Of Mind*. en. Simon & Schuster. ISBN: 978-0-671-65713-0 (cit. on p. 13).

Mitchell, Melanie (2006). "Complex systems: Network thinking". In: *Artificial Intelligence* 170.18, pp. 1194–1212. ISSN: 0004-3702 (cit. on p. 39).

Molenaar, Peter C. M., Karl M. Newell, and Richard M. Lerner (2013). *Handbook of Developmental Systems Theory and Methodology*. 1st ed. The Guilford Press. ISBN: 1-60918-509-9 978-1-60918-509-1 (cit. on p. 38).

Nakamoto, Satoshi (2008). "Bitcoin: A peer-to-peer electronic cash system". In: *Consulted* 1.2012, p. 28 (cit. on pp. 100, 108).

Nayebi, Aran (2014). "Practical intractability: A critique of the hypercomputation movement". In: *Minds and Machines* 24.3, pp. 275–305 (cit. on pp. 91–93, 98, 125).

Neumann, John von (June 1945). *First Draft of a Report on the EDVAC*. English. Tech. rep. W–670–ORD–4926. Moore School of Electrical Engineering University of Pennsylvania (cit. on p. 125).

Newell, Allen, J.C. Shaw, and H.A. Simon (Oct. 1958). "Chess-Playing Programs and the Problem of Complexity". In: *IBM Journal of Research and Development* 2.4, pp. 320–335. ISSN: 0018-8646 (cit. on pp. 192, 193).

Nilsson, Nils J. (2005). "Human-Level Artificial Intelligence? Be Serious!" In: *AI Magazine* 26, pp. 68–75 (cit. on p. 176).

Noorman, Job et al. (July 2017). "Sancus 2.0: A Low-Cost Security Architecture for IoT Devices". In: *ACM Trans. Priv. Secur.* 20.3, 7:1–7:33. ISSN: 2471-2566 (cit. on pp. 181, 182).

Odling-Smee, F. John (1988). "Niche-constructing phenotypes". In: *The role of behavior in evolution*. Ed. by H.C. Plotkin. Cambridge, MA, US: The MIT Press, pp. 77–132 (cit. on p. 36).

Odling-Smee, F. John, Kevin N. Laland, and Marcus W. Feldman (Feb. 2013). *Niche Construction: The Neglected Process in Evolution (MPB-37)*. en. Princeton University Press. ISBN: 1-4008-4726-5 (cit. on pp. 36, 37).

Oizumi, Masafumi, Larissa Albantakis, and Giulio Tononi (2014). "From the Phenomenology to the Mechanisms of Consciousness: Integrated Information Theory 3.0". In: *PLOS Computational Biology* 10.5, pp. 1–25 (cit. on p. 53).

Oyama, Susan (Mar. 2000). *The Ontogeny of Information: Developmental Systems and Evolution*. English. 2 edition. Durham, N.C.: Duke University Press Books. ISBN: 978-0-8223-2466-9 (cit. on pp. 38, 60, 65).

Oyama, Susan, Paul E. Griffiths, and Russell D. Gray, eds. (2001). *Cycles of Contingency - Developmental Systems and Evolution*. English. MIT Press (cit. on p. 37).

Pagallo, Ugo (2018). "Apples, oranges, robots: four misunderstandings in today's debate on the legal status of AI systems". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376.2133, p. 20180168 (cit. on p. 176).

Page, Lawrence et al. (1999). *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab (cit. on p. 138).

Piaget, Jean (Oct. 1971). *Genetic Epistemology*. English. Trans. by Eleanor Duckworth. New York: W W Norton & Co Inc. ISBN: 978-0-393-00596-7 (cit. on p. 84).

— (2004). *Genetic epistemology*. Russian. 5th ed. Saint Petersburg: Piter. ISBN: 5-318-00032-0 (cit. on pp. 53, 55).

Piccinini, Gualtiero (2011). "The physical Church–Turing thesis: Modest or bold?" In: *The British Journal for the Philosophy of Science* 62.4, pp. 733–769 (cit. on pp. 93, 94).

— (2015). "Pancomputationalism". In: *Physical Computation: A Mechanistic Account*. Oxford: Oxford University Press. ISBN: 978-0-19-965885-5 (cit. on p. 91).

— (2017). "Computation in Physical Systems". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2017. Metaphysics Research Lab, Stanford University (cit. on p. 93).

Pinker, Steven (May 2010). "The cognitive niche: Coevolution of intelligence, sociality, and language". en. In: *Proceedings of the National Academy of Sciences* 107.Supplement 2, pp. 8993–8999. ISSN: 0027-8424, 1091-6490 (cit. on p. 37).

Pintea, Camelia-Mihaela (2014). *Advances in Bio-inspired Computing for Combinatorial Optimization Problems*. 1st ed. Intelligent Systems Reference Library 57. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-40178-7 978-3-642-40179-4 (cit. on p. 48).

Powers, W. T., R. K. Clark, and R. L. Mc Farland (1960). "A General Feedback Theory of Human Behavior: Part I". In: *Perceptual and Motor Skills* 11.1, pp. 71–88 (cit. on p. 78).

Powers, W. T., R. K. Clark, and R. L. McFarland (1960). "A General Feedback Theory of Human Behavior: Part II". In: *Perceptual and Motor Skills* 11.3, pp. 309–323 (cit. on pp. 43, 44, 46, 47).

Powers, William T (1973). *Behavior: The control of perception*. Aldine Chicago (cit. on p. 24).

Powers, William T. (May 2016). *Perceptual Control Theory: An Overview of the Third Grand Theory in Psychology*. English. Ed. by Dag Forssell. 2016 edition. S.l.: Living Control Systems Publishing. ISBN: 978-1-938090-12-7 (cit. on pp. 43, 45).

Pritchett, Dan (2008). "BASE: An ACID alternative". In: *ACMQueue* 6.3, pp. 48–55 (cit. on p. 132).

Prokopenko, Mikhail (2009). "Guided self-organization". English. In: *HFSP Journal* 3.5, pp. 287–289. ISSN: 1955-2068 (cit. on p. 49).

— ed. (2014). *Guided Self-Organization: Inception*. 1st ed. Emergence, Complexity and Computation 9. Springer. ISBN: 978-3-642-53733-2 978-3-642-53734-9 (cit. on p. 49).

PubNub (June 2017). *Moving the Cloud to the Edge*. English. Blog (cit. on p. 186).

Raval, Siraj (2016). *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. English. 1st ed. Computers / finance. O'Reilly Media. ISBN: 1-4919-2454-3 978-1-4919-2454-9 (cit. on p. 187).

Raynal, Michel (June 2013). *Distributed Algorithms for Message-Passing Systems*. en. Springer Science & Business Media. ISBN: 978-3-642-38123-2 (cit. on pp. 102, 105, 112, 123).

Ren, Wei and Yongcan Cao (2011). *Distributed Coordination of Multi-agent Networks: Emergent Problems, Models, and Issues*. Communications and Control Engineering. Springer (cit. on p. 41).

Robinson, Ian, Jim Weber, and Efil Eifrem (2015). *Graph Databases: New Opportunities For Connected Data*. 2nd ed. (cit. on pp. 130, 132, 157).

Rodriguez, Marko (2011). *Graphs, Brains, and Gremlin* (cit. on p. 49).

— (Oct. 2012). *A Solution to the Supernode Problem*. English. Blog (cit. on pp. 139, 141).

— (July 2015a). *Tales from the TinkerPop*. English. Blog (cit. on p. 135).

Rodriguez, Marko A. (Oct. 2008a). "Grammar-based random walkers in semantic networks". In: *Knowledge-Based Systems* 21.7, pp. 727–739. ISSN: 0950-7051 (cit. on pp. 132, 134, 135, 137).

— (Apr. 2008b). "Mapping Semantic Networks to Undirected Networks". In: *arXiv:0804.0277 [cs]*. arXiv: 0804.0277 (cit. on p. 132).

— (2010). "General-Purpose Computing on a Semantic Network Substrate". In: *Emergent Web Intelligence: Advanced Semantic Technologies*. Ed. by Youakim Badr et al. London: Springer London, pp. 57–102. ISBN: 978-1-84996-077-9 (cit. on p. 101).

— (2015b). "The Gremlin Graph Traversal Machine and Language". In: *Proceedings of the 15th Symposium on Database Programming Languages*. DBPL 2015. New York, NY, USA: ACM, pp. 1–10. ISBN: 978-1-4503-3902-5 (cit. on pp. 133, 134, 139).

— (June 2017). "Open Problems in the Universal Graph Theory". In: San Francisco, CA, USA: Zenodo (cit. on p. 130).

Rodriguez, Marko A. and Johan Bollen (May 2007). "Modeling Computations in a Semantic Network". In: *arXiv:0706.0022 [cs]*. arXiv: 0706.0022 (cit. on p. 133).

Rodriguez, Marko A. and Joe Geldart (Oct. 2008). "An Evidential Path Logic for Multi-Relational Networks". In: *arXiv:0810.1481 [cs]*. arXiv: 0810.1481 (cit. on p. 138).

Rodriguez, Marko A. and Peter Neubauer (June 2010). "Constructions from Dots and Lines". In: *arXiv:1006.2361 [cs]*. arXiv: 1006.2361 (cit. on pp. 41, 43).

Rodriguez, Marko A and Peter Neubauer (2011). "A path algebra for multi-relational graphs". In: *Data Engineering Workshops (ICDEW), 2011 IEEE 27th International Conference on*. IEEE, pp. 128–131 (cit. on pp. 135, 138).

Rodriguez, Marko A., Alberto Pepe, and Joshua Shinavier (June 2010). "The Dilated Triple". In: *arXiv:1006.1080 [cs]*. arXiv: 1006.1080 (cit. on p. 132).

Rodriguez, Marko A. and Joshua Shinavier (Jan. 2010). "Exposing Multi-Relational Networks to Single-Relational Network Analysis Algorithms". In: *Journal of Informetrics* 4.1. arXiv: 0806.2274, pp. 29–41. ISSN: 17511577 (cit. on p. 135).

Rodriguez Marko, Marko A. (2016a). *A Gremlin Implementation of the Gremlin Traversal Machine* (cit. on p. 142).

— (Sept. 2016b). *Gremlin's Time Machine* (cit. on pp. 140, 143).

Rodríguez, José María Álvarez, José Emilio Labra Gayo, and Patricia Ordoñez De Pablos (Jan. 2013). "ONTOSPREAD: A Framework for Supporting the Activation of Concepts in Graph-Based Structures through the Spreading Activation Technique". In: *Information Systems, E-learning, and Knowledge Management Research*. Ed. by Miltiadis D. Lytras et al. Communications in Computer and Information Science 278. Springer Berlin Heidelberg, pp. 454–459. ISBN: 978-3-642-35878-4 978-3-642-35879-1 (cit. on p. 49).

Ruohonen, Keijo (2013). "Graph Theory". English. Tampere University of Technology, Finland (cit. on p. 42).

Russell, Matthew A. (2019). *Mining the social web: data mining Facebook, Twitter, LinkedIn, Instagram, GitHub, and more*. 3rd ed. O'Reilly Media. ISBN: 978-1-4919-8504-5 (cit. on p. 133).

Sandberg, Oscar (2005). "Searching in a small world". English. PhD Thesis. Goteborg, Sweden: Chalmers University of Technology and Goteborg University (cit. on p. 140).

Sayama, Hiroki (2014). "Guiding Designs of Self-Organizing Swarms: Interactive and Automated Approaches". In: *Guided Self-Organization: Inception*. Ed. by Mikhail Prokopenko. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 365–387. ISBN: 978-3-642-53734-9 (cit. on p. 117).

Sayama, Hiroki et al. (2013). "Modeling complex systems with adaptive networks". In: *Computers & Mathematics with Applications* 65.10, pp. 1645–1664. ISSN: 0898-1221 (cit. on p. 41).

Schacter, Daniel L., Daniel T. Gilbert, and Daniel M. Wegner (Dec. 2010). *Psychology*. en. Worth Publishers. ISBN: 978-1-4292-3719-2 (cit. on p. 53).

Schneider, Nathan (Apr. 2014). *Code your own utopia: Meet Ethereum, bitcoin's most ambitious successor*. English (cit. on p. 187).

Scott-Phillips, Thomas C. et al. (May 2014). "The niche construction perspective: a critical appraisal". eng. In: *Evolution; International Journal of Organic Evolution* 68.5, pp. 1231–1243. ISSN: 1558-5646 (cit. on p. 37).

Shadbolt, Nigel, Tim Berners-Lee, and Wendy Hall (2006). "The semantic web revisited". In: *IEEE intelligent systems* 21.3, pp. 96–101 (cit. on p. 131).

Shang, Zechao and Jeffrey Xu Yu (Oct. 2014). "Auto-approximation of Graph Computing". In: *Proc. VLDB Endow.* 7.14, pp. 1833–1844. ISSN: 2150-8097 (cit. on p. 138).

Shannon, C. E. (July 1948). "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3, pp. 379–423. ISSN: 0005-8580 (cit. on pp. 64, 65).

Shannon, Claude E (1950). "Programming a computer for playing chess". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41.314, pp. 256–275 (cit. on p. 192).

Simon, H. A. (1990). "Invariants of human behavior". eng. In: *Annual Review of Psychology* 41, pp. 1–19. ISSN: 0066-4308 (cit. on p. 50).

Simon, Herbert A. (1962). "The architecture of complexity". In: *Proceedings of the American Philosophical Society*, pp. 467–482 (cit. on pp. 20–23, 27, 28, 69, 77, 113).

Simon, Herbert Alexander (1982). *Models of Bounded Rationality: Empirically grounded economic reason*. en. Google-Books-ID: 9CiwU28z6WQC. MIT Press. ISBN: 978-0-262-19372-6 (cit. on p. 31).

Simondon, Gilbert (1980). "On the mode of existence of technical objects". English. University of Western Ontario (cit. on pp. 63, 176, 177, 179, 180, 189).

— (1992). "The Genesis of the Individual". In: *Incorporations*. Ed. by Jonathan Crary and Sanford Kwinter. New York: Zone, pp. 297–319. ISBN: 0942299299 (paper) 0942299302 (cloth) (cit. on pp. 6, 63, 176).

— (Nov. 2005). *L'individuation à la lumière des notions de forme et d'information [Individuation in light of the notions of form and information]*. Français. Grenoble: Editions Jérôme Millon. ISBN: 978-2-84137-181-5 (cit. on pp. 6, 63).

— (2009). "The Position of the Problem of Ontogenesis". In: *Parrhesia* 7, pp. 4–16 (cit. on pp. 38, 63, 77).

Sims, Chris R. (July 2016). "Rate–distortion theory and human perception". In: *Cognition* 152.Supplement C, pp. 181–198. ISSN: 0010-0277 (cit. on pp. 33, 51, 52).

Smart, John M. (Nov. 2015). "Humanity Rising: Why Evolutionary Developmentalism Will Inherit the Future". In: *World Futures Review* 7.2-3, pp. 116–130. ISSN: 1946-7567 (cit. on p. 35).

— (2017). *What is Evolutionary Development?* English. Alpha version (cit. on p. 35).

Smelser, Neil J and Richard Swedberg (2005). "Introducing economic sociology". In: *The handbook of economic sociology* 2, pp. 3–25 (cit. on p. 150).

Smith, Daniel and John Protevi (2013). "{Gilles} {Deleuze}". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2013 (cit. on p. 68).

Smith, J. Maynard et al. (1985). "Developmental Constraints and Evolution: A Perspective from the Mountain Lake Conference on Development and Evolution". In: *The Quarterly Review of Biology* 60.3, pp. 265–287 (cit. on p. 35).

Solaiman, SM (2017). "Legal personality of robots, corporations, idols and chimpanzees: a quest for legitimacy". In: *Artificial Intelligence and Law* 25.2, pp. 155–179 (cit. on p. 176).

Solum, Lawrence B (1991). "Legal personhood for artificial intelligences". In: *NCL Rev.* 70, p. 1231 (cit. on p. 176).

Stannett, Mike (2004). "Hypercomputational Models". In: *Alan Turing: Life and Legacy of a Great Thinker*. Ed. by Christof Teuscher. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 135–157. ISBN: 978-3-662-05642-4 (cit. on pp. 91, 92).

Steels, Luc (2008). "The Symbol Grounding Problem has Been Solved. So What's Next". In: *Symbols and Embodiment: Debates on Meaning and Cognition*. Ed. by Manuel de Vega, Arthur M. Glenberg, and Arthur C. Graesser. Oxford University Press, pp. 223–244 (cit. on p. 89).

— (May 2015). *The Talking Heads experiment: Origins of words and meanings*. en. Google-Books-ID: jtEsCQAAQBAJ. Language Science Press. ISBN: 978-3-944675-42-8 (cit. on p. 89).

Stenhouse, David (1974). *The evolution of intelligence: A general theory and some of its implications*. Barnes & Noble Books (cit. on p. 76).

Stotz, Karola (Dec. 2010). "Human nature and cognitive–developmental niche construction". en. In: *Phenomenology and the Cognitive Sciences* 9.4, pp. 483–501. ISSN: 1568-7759, 1572-8676 (cit. on p. 37).

Stützle, Thomas and Marco Dorigo (1999). "ACO algorithms for the traveling salesman problem". In: *Evolutionary algorithms in engineering and computer science*, pp. 163–183 (cit. on pp. 48, 116).

Summers, Frank (1994). *Object Relations Theories and Psychopathology: A Comprehensive Text*. en. Analytic Press. ISBN: 978-0-88163-155-5 (cit. on p. 54).

Swan, Melanie (Feb. 2015). *Blockchain: Blueprint for a New Economy*. English. 1 edition. O'Reilly Media. ISBN: 978-1-4919-2049-7 (cit. on p. 164).

Taleb, Nassim et al. (2012). *A New Heuristic Measure of Fragility and Tail Risks: Application to Stress Testing*. IMF working paper. International monetary fund (IMF) (cit. on p. 40).

Taleb, Nassim Nicholas (Nov. 2012). *Antifragile: Things That Gain from Disorder*. Random House. ISBN: 1-4000-6782-0 (cit. on p. 40).

Taleb, Nassim Nicholas and Raphael Douady (2013). "Mathematical definition, mapping, and detection of (anti) fragility". In: *Quantitative Finance* 13.11, pp. 1677–1689 (cit. on p. 40).

Taylor, M. M. (June 1999). "Editorial: Perceptual Control Theory and its application". In: *International Journal of Human-Computer Studies* 50.6, pp. 433–444. ISSN: 1071-5819 (cit. on p. 43).

Thagard, Paul (2002). *Coherence in Thought and Action*. en. MIT Press. ISBN: 978-0-262-70092-4 (cit. on pp. 52, 80).

Thrun, Sebastian, Dieter Fox, and Wolfram Burgard (2005). *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. ISBN: 0-262-20162-3 978-0-262-20162-9 (cit. on p. 18).

Todd, Peter M. and Gerd Gigerenzer (2012). *Ecological Rationality: Intelligence in the World*. 1st ed. Evolution and Cognition. Oxford University Press. ISBN: 0-19-531544-8 978-0-19-531544-8 (cit. on p. 50).

Tomasini, Marcello (2015). "An introduction to multilayer networks". In: *BioComplex Laboratory, Florida Institute of Technology, Melbourne, USA*, pp. 1–14 (cit. on p. 42).

Tononi, Giulio (Nov. 2004). "An information integration theory of consciousness". en. In: *BMC Neuroscience* 5.1, p. 42. ISSN: 1471-2202 (cit. on pp. 24, 53, 70).

— (Dec. 2008). "Consciousness as Integrated Information: a Provisional Manifesto". en. In: *The Biological Bulletin* 215.3, pp. 216–242. ISSN: 0006-3185, 1939-8697 (cit. on p. 53).

Tononi, Giullo, Olaf Sporns, and Gerald M. Edelman (July 1992). "Reentry and the Problem of Integrating Multiple Cortical Areas: Simulation of Dynamic Integration in the Visual System". en. In: *Cerebral Cortex* 2.4, pp. 310–335. ISSN: 1047-3211, 1460-2199 (cit. on pp. 80, 81).

Trewavas, Anthony (July 2003). "Aspects of Plant Intelligence". In: *Annals of Botany* 92.1, pp. 1–20. ISSN: 0305-7364 (cit. on p. 76).

— (2004). "Aspects of Plant Intelligence: an Answer to Firn". In: *Annals of Botany* 93.4, pp. 353–357 (cit. on p. 76).

Tuohy, Shane et al. (2015). "Intra-vehicle networks: A review". In: *IEEE Transactions on Intelligent Transportation Systems* 16.2, pp. 534–545 (cit. on p. 179).

Turchin, Valentin and Cliff Joslyn (1993). "The Metasystem Transition". In: *Principia Cybernetica Web*. Ed. by Valentin Turchin and Cliff Joslyn. Brussels: Principia Cybernetica (cit. on p. 26).

Turchin, Valentin F. (1986). "The Concept of a Supercompiler". In: *ACM Transactions on Programming Languages and Systems* 8, pp. 292–325 (cit. on p. 101).

Turchin, Valentin Fedorovich (Oct. 1977). *The Phenomenon of Science: A Cybernetic Approach to Human Evolution*. English. 1st edition. New York: Columbia Univ Pr. ISBN: 978-0-231-03983-3 (cit. on pp. 25, 26).

Turing, A. M. (Oct. 1950). "Computing Machinery and Intelligence". In: *Mind*. New Series 59.236, pp. 433–460. ISSN: 0026-4423 (cit. on p. 11).

Turing, Alan (1938). *Systems of Logic Based on Ordinals*. London Mathematical Society (cit. on pp. 92, 101).

Turing, Alan M (1937). "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London mathematical society* 2.1, pp. 230–265 (cit. on pp. 48, 91, 92, 95, 97, 98, 101, 102, 117, 121, 125, 145, 172, 187, 190).

— (1948). "Intelligent Machinery". In: *Machine Intelligence 5*. Ed. by B Meltzer and D Michie. Edinburgh: Edinburgh University Press (cit. on pp. 92, 95–97, 101, 103, 117, 118, 145, 186, 190).

Tëmkin, Ilya and Niles Eldredge (2015). "Networks and Hierarchies: Approaching Complexity in Evolutionary Theory". en. In: *Macroevolution*. Ed. by Emanuele Serrelli and Nathalie Gontier. Interdisciplinary Evolution Research. Springer, Cham, pp. 183–226. ISBN: 978-3-319-15044-4 978-3-319-15045-1 (cit. on pp. 19, 23).

Van Bulck, Jo, Jan Tobias Muhlberg, and Frank Piessens (Dec. 2017). "Efficient component Authentication and Software Isolation for Automotive Control Networks". In: Submitted version. San Juan, Puerto RIco, USA (cit. on pp. 179, 182).

Van Bulck, Jo, Jan Tobias Mühlberg, and Frank Piessens (2017). "VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks". In: *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACSAC 2017. New York, NY, USA: ACM, pp. 225–237. ISBN: 978-1-4503-5345-8 (cit. on p. 179).

Veitas, Viktoras and David Weinbaum (2015). "A World of Views". In: *The End of the Beginning: Life, Society and Economy on the Brink of the Singularity*. Ed. by Ben Goertzel and Ted Goertzel. Accepted for publication. (cit. on pp. 40, 56, 192).

— (2017). "Living Cognitive Society: A 'digital' World of Views". In: *Technological Forecasting and Social Change* 114, pp. 16 –26. ISSN: 0040-1625 (cit. on pp. 23, 29, 63, 79, 105).

Veitas, Viktoras et al. (July 2015). *Governing the Future's Power System: toward a method of guided self-organisation for the Smart Grid*. English. Working Paper. Brussels: Global Brain Institute, CLEA-VUB (cit. on p. 184).

Veitas, Viktoras Kabir and Simon Delaere (Feb. 2018a). *In-vehicle data recording, storage and access management in autonomous vehicles*. English. Technical report. Brussels: imec-SMIT-VUB, p. 21 (cit. on pp. 177–179).

— (Feb. 2018b). *Policy Scan and Technology Strategy Design methodology*. English. Technical report. Brussels: imec-SMIT-VUB, p. 20 (cit. on p. 178).

Venkatasubramanian, Nalini and Carolyn Talcott (1995). "Reasoning About Meta Level Activities in Open Distributed Systems". In: *Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing*. PODC '95. New York, NY, USA: ACM, pp. 144–152. ISBN: 0-89791-710-3 (cit. on p. 129).

Vidal, Clément (June 2008). *What is a worldview?* Published in Dutch. (cit. on p. 56).

Vidal, Clément and Steven J. Dick (2014). *The beginning and the end : the meaning of life in a cosmological perspective*. 2014th ed. Frontiers collection. Springer. ISBN: 3-319-05061-3 978-3-319-05061-4 978-3-319-05062-1 3-319-05062-1 (cit. on pp. 47, 56).

Vitanyi, P. M.B. (2009). "Turing machine". In: *Scholarpedia* 4.3, p. 6240 (cit. on p. 91).

Walker, Brian et al. (2004). "Resilience, adaptability and transformability in social–ecological systems". In: *Ecology and society* 9.2, p. 5 (cit. on pp. 71–73).

Wallace, Arthur (Feb. 2002). "The emerging conceptual framework of evolutionary developmental biology". In: *Nature* 415, p. 757 (cit. on p. 35).

Wang, Pei (Dec. 2005). "Experience-grounded Semantics: A Theory for Intelligent Systems". In: *Cogn. Syst. Res.* 6.4, pp. 282–302. ISSN: 1389-0417 (cit. on p. 137).

Watts, Duncan J. and Steven H. Strogatz (June 1998). "Collective dynamics of łqsmall-world\r q networks". In: *Nature* 393, p. 440 (cit. on p. 168).

Weeratunga, Kamal and Andrew Somers (June 2015). *Connected Vehicles: Are We Ready?* English. Internal Report. East Perth, Western Australia: Main Roads WA, p. 52 (cit. on p. 178).

Wegner, Peter (1998). "Interactive foundations of computing". In: *Theoretical computer science* 192.2, pp. 315–351 (cit. on p. 97).

Weinbaum, David R (2013). "A Framework for Scalable Cognition". In: *Proceedings of the European Conference on Complex Systems 2012*. Springer, pp. 559–567 (cit. on pp. 30, 34, 66, 73).

Weinbaum, David R. (2015). "Complexity and the Philosophy of Becoming". In: *Foundations of Science* 20.3, pp. 283–322 (cit. on p. 76).

Weinbaum, David R. (Weaver) (Mar. 2018). "Open-ended intelligence". English. PhD Thesis. Brussels: Vrije Universiteit Brussel (cit. on pp. 3, 15, 48, 53, 60, 63, 87, 194).

Weinbaum, David (Weaver) and V. Veitas (Jan. 2017a). "Synthetic cognitive development". en. In: *The European Physical Journal Special Topics* 226.2, pp. 243–268. ISSN: 1951-6355, 1951-6401 (cit. on pp. 24, 53, 55, 56, 63, 71, 73, 74, 77–79, 82, 84, 85).

Weinbaum, David (Weaver) and Viktoras Veitas (Mar. 2017b). "Open ended intelligence: the individuation of intelligent agents". In: *Journal of Experimental & Theoretical Artificial Intelligence* 29.2, pp. 371–396. ISSN: 0952-813X (cit. on pp. 7, 15, 34, 53, 56, 66).

Weiss, Gerhard (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. en. MIT Press. ISBN: 978-0-262-73131-7 (cit. on pp. 31, 40).

Wickham, Hadley and others (2011). "The split-apply-combine strategy for data analysis". In: *Journal of Statistical Software* 40.1, pp. 1–29 (cit. on p. 109).

Wiener, Norbert (1950). *The Human Use of Human Beings: Cybernetics and Society*. en. Boston: Houghton-Mifflin (cit. on p. 43).

Yampolskiy, Roman V. (2015). *Artificial Superintelligence: A Futuristic Approach*. Chapman and Hall / CRC. ISBN: 1-4822-3443-2 978-1-4822-3443-5 (cit. on p. 62).

Zenil, Hector et al. (Nov. 2012). "Life as Thermodynamic Evidence of Algorithmic Structure in Natural Environments". In: *Entropy* 14.11, pp. 2173–2191. ISSN: 1099-4300 (cit. on pp. 60, 62).