FACULTY OF SCIENCE AND TECHNOLOGY

A THESIS

submitted in partial fulfillment
of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

by

Michele Bottone

Agoric Computation: Trust and Cyber-Physical Systems

Middlesex University London,

Department of Computer Science,

The Burroughs, London NW4 4BT, United Kingdom

MAY 2018

Declaration

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Acknowledgements

A thesis is by definition a solitary endeavour, yet academic research is primarily a social enterprise, and owes much to interactions with a long list of people. The work featured in this thesis makes no exception. In addition to the usual close suspects, which I have been thanking for much of my adult life and shall remain anonymous, I would like to give heartfelt thanks to the following: my Director of Studies, Professor Franco Raimondi, for his guidance, willingness to discuss anything in relaxed settings, and tireless administrative duties; my other supervisor, Professor Raja Nagarajan, who encouraged me to complete this doctorate alongside my day job, and suggested attending BCTCS where more or less the seeds were planted; several collaborators, conference co-attendants and anonymous referees for improving my writing; my managers Martin, Tony, Balbir and Miltos for funding my PhD; and my office colleagues for countless lunches, patience, time spent together, and random sanity-preserving discussions on a wide range of subjects, among them Giuseppe, Kelly, Nikos, Taloue, Florian, Barny, Alessandro, Aga, Michael, Lorenzo, Matteo, Elisabetta, K and Letti.

Contents

Abstract

In the past two decades advances in miniaturisation and economies of scale have led to the emergence of billions of connected components that have provided both a spur and a blueprint for the development of smart products acting in specialised environments which are uniquely identifiable, localisable, and capable of autonomy. Adopting the computational perspective of multi-agent systems (MAS) as a technological abstraction married with the engineering perspective of cyber-physical systems (CPS) has provided fertile ground for designing, developing and deploying software applications in smart automated context such as manufacturing, power grids, avionics, healthcare and logistics, capable of being decentralised, intelligent, reconfigurable, modular, flexible, robust, adaptive and responsive. Current agent technologies are, however, ill suited for information-based environments, making it difficult to formalise and implement multi-agent systems based on inherently dynamical functional concepts such as trust and reliability, which present special challenges when scaling from small to large systems of agents. To overcome such challenges, it is useful to adopt a unified approach which we term *agoric computation*, integrating logical, mathematical and programming concepts towards the development of agent-based solutions based on recursive, compositional principles, where smaller systems feed via directed information flows into larger hierarchical systems that define their global environment. Considering information as an integral part of the environment naturally defines a web of operations where components of a systems are wired in some way and each set of inputs and outputs are allowed to carry some value. These operations are stateless abstractions and procedures that act on some stateful cells that cumulate partial information, and it is possible to compose such abstractions into higher-level ones, using a publish-and-subscribe interaction model that keeps track of update messages between abstractions and values in the data. In this thesis we review the logical and mathematical basis of such abstractions and take steps towards the software implementation of agoric modelling as a framework for simulation and verification of the reliability of increasingly complex systems, and report on experimental results related to a few select applications, such as stigmergic interaction in mobile robotics, integrating raw data into agent perceptions, trust and trustworthiness in orchestrated open systems, computing the epistemic cost of trust when reasoning in networks of agents seeded with contradictory information, and trust models for distributed ledgers in the Internet of Things (IoT); and provide a roadmap for future developments of our research.

Chapter 1

Introduction

What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle. Marvin Minsky, The Society of Mind, p. 308

1.1 Motivation

A wide class of problems in computer science concerns the *flow of information* in a collection of autonomous yet interconnected entities, which may be of fixed size or open ended. Such interacting components, ubiquitous in nature, language and society, may scale from a handful to several billions, and there is a complex interplay between local characteristics and higher level functions defined on the components or the system as a whole (to achieve its purpose), from which some global behaviour arises. One can usually express the overall capabilities of the system as "the whole is more than the sum of the parts" or as an "interactive system". Such a system is a collection of subsystems, whose behaviour emerges from local interaction and self-organisation. Local characteristics describe the data or 'raw facts', and it is often the case that they can be described *succinctly* and *dynamically* in time and/or space by the use of appropriate variables (also called factors, inputs, features, or signals), which greatly reduce the dimensionality of a problem to be solved, and render the resultant system fault-tolerant despite being composed of groups of different, error-prone actors. There are various descriptions and nomenclatures for such systems, but one striking feature that describes successful, adaptive systems that make sense of data in a noisy world is their information processing, which allows a system to extract relevant information more efficiently despite the additional cost of doing the information processing. Understanding how the bottlenecks that prevent large scale, efficient synthesis and feedback of such systems – such as the well-known phenomenon of combinatorial explosion as the number of component increases, and the presence of noise – are eased in real-world systems therefore motivates the study of collective computation. Dimensionality reduction, in particular, helps manage combinatorial explosion, gives a design cue to implementing scalable software systems and has been the main motivation of this thesis. Indeed, along each of these dimensions the internal mechanisms of the system continuously process and integrate a stream of incoming data – or inputs – into readily available representations – or outputs – which in turn become more data of the system. From the point of view of an external observer or modeller, one can use a formalisation either in a forward-looking manner (i.e. for prediction and simulation of the future states) or backward-looking manner (i.e. for inference and testing of parameters and other local and global characteristics). Situations where this interplay between interaction and information occurs are ubiquitous, shaped by physical constraints, institutions, norms, evolution and social dynamics [Goldin *et al.* 2006] and when viewed as abstractions or computational processes, have the potential of serving as a model of computation where the focus is on the dataflow or constraint propagation with interesting properties that mirror in some way both naturally occurring phenomena and philosophical theories about the logic of uncertainty and information – such as bayesian statistics, relational networks, reinforcement learning.

Biological systems, and in particular sentient, social beings and cognitive actors such as animals and humans, are able to solve problems concerning partial information easily, thanks to favourable aspects such as reactivity, goal-oriented behaviour, narrow and global optimisation capabilities in the face of constraints, ability to synthesise disparate attributes, resilience to faults and contradictions, and flexible strategy selection. It is likely that this requires not just reactive behaviour but pro-active behaviour in the face of hypothetical scenarios, that makes use of abstract, conceptual and symbolic reasoning in the form of patterns and rules as well as hard, numerical data, and the work presented in this thesis aims to make some definite progress towards realising a logical and computational framework for applications where information exchange takes place. While progress has been made in understanding collective computation [Flack 2017] it remains difficult to formalise and engineer such system in the face of the complexities required to build such systems at scale.

1.2 Multi-Agent Systems

One perspective that has recently been popular in the computer science literature is that of multi-agent systems (MAS), which have surfaced in academia [Wooldridge 2003], industrial applications [Vrba 2013] and general public discourse [Shanahan 2015] as an essential addition to the computer science toolbox. At a very basic level, multi-agent systems consist of a dynamic collection of agents with their own beliefs and goals, that interact with each other in a situated, shared environment, whose nature may be physical or virtual, discrete or continuous.

The environment, in turn, allows actions and feedback – including reasoning and communication – to be performed, such that the system meets some global requirements towards solving problems. These specify further goals that can be met by agents acting co-operatively, competitively or both to discover patterns and solutions, given available resources and constraints. In multi-agent systems, coordination is required because of dependencies, constraints, and partial information. An interesting aspect of these is how a system reasons and decides, via commitments (a special kind of goal) and their associated conventions, norms and deliberations [Jennings 2000]. For these reasons, MAS can be built up from scratch using modules where the programs give an easily customisable and testable model for these systems. Adopting the MAS perspective of software agency allows us to naturally model, design, implement and reason about several important problems in artificial intelligence through their shared characteristics of *autonomy* (the agents are at least partially independent or self-aware), *locality* (no agent has a full global view of the environment, or it is difficult to make use of it in a complex system) and *decentralisation* (there is no designated controlling agent) at a sufficiently high level of abstraction. In this context, intelligence is an emergent property of a system and its inherent diversity and interaction, rather than a fixed property of individual agents. In synthesis, an *agent* is usually considered to be an entity with one or more of the following properties: *autonomy, social ability, reactivity, pro-activity* [Wooldridge 2003]. A MAS could also be easily characterised by means of its *Beliefs*, *Desires*, and *Intentions (BDI).*

1.2.1 BDI Logics

A popular theory of agency is the *belief-desire-intention* philosophy of [Bratmann 1999], based on the Aristotelian concept of practical reasoning by humans, or how they rationally make decisions and take actions as a result of mental states. The BDI architecture has proved to be popular in the specification, programming, and implementation of artificial agent technologies [Rao & Georgeff 1998a]. Loosely, within a reasoning cycle, the agent has *beliefs*, based on what it perceives and communicates with other agents; beliefs can produce *desires*, intended as states of the world that the agent wants to achieve; the agent deliberates on its desires and decides to commit to some; desires to which the agent is committed become *intentions*, to satisfy which the agent executes *plans* that lead to action. The behaviour of the agent (i.e., its actions) is thus explained or caused by what it intends (i.e., the desires it decided to pursue). Ideally, within the BDI architecture, agents should react to changes in its environment as soon as possible while keeping its proactive (i.e., desired-action-oriented) behaviour. BDI logics provide the backbone of a readily formalisable rule-based system that be integrated with message-passing and an agent communication infrastructure, is amenable to customisable implementation and features prominently in current research as a rapid prototyping tool.

1.3 Cyber-Physical Systems

An emerging and fascinating area of research is that of Cyber-Physical Systems (CPS), namely orchestrations of computers and physical components in a dynamic environment. Computing devices and networks are used to monitor and control the physical parts, possibly with tightly coupled feedback loops where computations continuously affect physical processes and vice versa. Applications of such systems are manifold [NIST] and include manufacturing and process control, autonomous automotive systems, medical devices, smart wearables, military systems, avionics and flight safety, air traffic control, e-health and assisted living, power generation and distribution, energy efficiency and conservation in buildings, water management systems, trains, access control and monitoring, livestock, physical and computational asset management, robotics and and ever-increasing plethora of emerging technologies driven by real-life physical and social processes. Designing such open systems requires an intimate understanding of the joint dynamics of computers, software, networks, and the underlying physical processes, where it is not enough to focus on expertise in either domain; by definition, this area is set apart by its focus on the intersection of physical and computational processes rather than their union [Derler *et al.* 2012, Akkaya 2016], and by their inherently open dataflow structure. This joint dynamics opens unique sets of challenges for researchers, modellers and designers which have largely to do with the composition of a wide range of heterogeneous components and the information they continually produce and that needs to be evaluated within the system by its computational units, and coming up with good models of interaction is essential going forward. CPS usually combine models and methods from electrical, biomedical, civil and mechanical, environmental, aeronautical and industrial engineering with the models, methods and frameworks of computer science and software engineering. It has been argued that such a heterogeneous set of models and methods do not combine easily and thus one should view CPS as a separate engineering discipline [Lee 2015], although the same can be said for any discipline where the focus is on solving problems. Often, as part of practical problem solving, CPS have to learn, adapt and evolve in real time, much in the way animals have been doing over very long time scales, and information processing and machine learning integrated with control mechanism of tools and data in CPS naturally blur the boundaries with artificial intelligence.

Several research directions have recently appeared which feature the broad theme of this thesis, namely connecting and orchestrating a potentially large number of variously 'smart' devices to humans and their environment in such a way that collective computation emerges. In particular, CPS has emerged as an umbrella term for several established and newer technologies, such as Industry 4.0, Machine2Machine, Ambient Intelligence/Intelligent Environment, Sensor Networks, and the Internet of Things/Everything (IoT/E), harking back to [Wiener 1948]'s pioneering "scientific study of control and communication in the animal and the machine". In particular, IoT systems are implementations of CPS where one emphasises the intensively networked nature of the constituent devices, leveraging the Internet and associated technologies to build a swarm of heterogeneous sensors and actuators that interacts with the environment and can dynamically solve problems in several domains of application and learn new rules and patterns of behaviour, and autonomously adapting to changes at run-time, under the implicit assumption that such systems are able to make real-time, autonomous decisions as new information is processed. One classic example is that of a home ambient and entertainment system that continually adjusts to the human activity in the background.

Despite the advances made in the past decades in the fields of artificial intelligence, software agents, robotics and sensor miniaturisation, fully realised autonomy is still some distance in the future. Desirable qualities of CPS should not only be *flexibility* and a certain degree of *autonomy*, but also quantifiable, social characteristics such as *trust* and *reliability* that enable the orchestrated system to operate smoothly and *mimimise risks to its integrity*, as well to triangulate and where possible preserve social and individual liberties such as privacy and free will by means of ethical considerations [(STOA) 2016].

In broader autonomous systems such as those encountered in open systems and CPS, the interplay between trust and reliability and the applicability of verification techniques [Baier & Katoen 2008] such as model checking for validating a trust model based entirely on logical inference has led to some paradoxes and malfunctioning when something unexpected happens. This is a particularly important problem in many systems, for instance for autopilot software and other emerging fields of application of trust modelling [Lahijanian & Kwiatkowska 2016]. In contrast to time-honed domains such as aviation and rail, another domain in which verification is important is that of autonomous cars, due to the different nature of possible paths, the proximity of hazards and the sheer volume of interactions with humans and other autonomous systems which require a holistic approach to reasoning in the presence of risk and uncertainty and the implementation of ethical systems. Data useful for information processing in CPS can be either abundant or scarce, lying in plain sight or obscured, and the right balance needs to be found to design them effectively and securely.

1.4 Towards Agoric Computation

In this Section, we set out the intellectual challenges that this thesis seeks to address, and introduce the methodology of Agoric Computation that forms the core of our contribution to the wider research area in its historical context. We also provide a discussion of the nature of information with examples and an initial review of related work.

1.4.1 Agoras and Information

When studying composite, open systems such as CPS, it is useful to use a marketplace metaphor for computation. Marketplaces are social constructs that have naturally emerged to deal with the logic of uncertainty and information, and provide a mechanism for filtering signal from noise and learning form data, even though these signals may be themselves hard to interpret taken in isolation. The aspects of the coordination, signalling and filtering problem that markets solve, i.e "how little the individual participants need to know in order to be able to take the right action" are well understood in economics [Hayek 1937, Hayek 1945, Otteson 2002] and evolutionary computational biology [Chastain *et al.* 2013]. We term such abstractions *Agoric Processes*[1] to underline the distributed, dynamic, pro-active and mixed computational-physical nature of such systems. Roughly speaking, within the confines of the title of the present work, *Agoric Computation* is an attempt to formalise the computational processes that arise in agoric environments viewed through the lens of MAS. Formally, for a fixed number N, a multi-agent system can be defined by a set of agents $\mathbf{Ag}$, which we take to be a finite set $\mathbf{Ag} \equiv [N] = \{1, 2, \ldots, N\}$, and denote individual agents by some index $a \in \mathbf{Ag}$.

- ▷ *Agoras* are locations where some parties meet and interact for the purposes of exchanging something of value, and to communicate signals about concepts, ideas, goods, and services, that involve information flows among diverse agents. People meet or communicate, and explicitly or implicitly exchange information from others, and this interaction may change some *state* carried by individuals. The fact that humans (and the machines that substitute them) do a lot of different things via interactions and working together emphasises the point that we are interested in looking at a variety of different rules [Goldin *et al.* 2006].
- ▷ *Information* refers to – possibly unobservable – attributes, such as energy, opinions, strategies, possessions, wealth, disease, spin, productivity, signals, and a "marketplace" is some kind of discrete *graph or network* structure connecting agents, i.e. describing their possibly changing relationships and local and global behaviour.

Agoric processes are defined along two levels of abstraction:

- ▷ The **geometry of interaction**, i.e. a "meeting" model of the network between connected agents, and/or
- ▷ An **update rule** – which is deterministic, but it can also be stochastic – that determines agents' responses, and provides the "information exchange" model.

The two levels of abstraction, the web of relations and the dynamic, are inseparable. Loosely speaking, an agoric process is a marketplace together with information flows, based on an iterated meet-and-exchange cycle for information propagation.

A blueprint for such computational processes was given by [Minsky 1986, 28.2], who envisaged a society of heterogeneous agents that compete and cooperate to produce mental capabilities, although again the idea is very old [Hayek 1952, Hebb 1949, Otteson 2002] and remains largely unimplemented in practice; see for instance

[1] In ancient Greece, the term $\alpha\gamma o\rho\grave{\alpha}$ designated both a meeting and a market place.

[Wright & Aubé 1998, Baum 2003, Zenil *et al.* 2017]. In the late 1980s and early 1990s there was considerable interest in distributed computational ecologies, software agents undertaking limited resource allocation, their interactions, strategies, and knowledge, and several attempts to deal with these systems were proposed that died out for lack of a coherent and powerful theory [Miller & Drexler 1988, Huberman & Hogg 1988, Huberman & Hogg 1995, Weiss 1995, Wellman 1999]. Like other ideas with a long and fertile intellectual history – such as deep convolutional learning networks – often significant progress is made when the right milieu and tools have been built up, and the past two decades have experienced a trickle of incremental progress that has recently led to an explosion of related techniques. Thus agents deal with a particular commodity called information. The great French mathematician René Thom called it a "semantic chameleon" [Thom 1983], something that easily changes in correspondence with the environment. In its different forms and appearances, the concept of information becomes closely related to notions of constraint, communication, control, data, form, education, knowledge, meaning, understanding, mental stimuli, pattern, perception, representation, and entropy. In its most restricted technical meaning, information is an ordered sequence of possibly changing symbols [Floridi 2010]. Of course, few concepts in science are as ubiquitous, elusive and pervasive as information, or lack thereof [Norretranders 1998]. It is pervasive, because all branches of science rely on some notion of information; on the other hand, it is also elusive because it is used to mean many different and contrasting things, ranging in a spectrum from the semantic to the technical [Soofi 1994]. In a semantic context, the term information is used in an intuitive, encompassing sense, and does not refer to a well defined logico-numerical quantity that can be used for measuring the extent of differentials or behavioural drifting arising as a result of changes in uncertain or deterministic states of nature. In a technical context, information is taken – for example by physicists and statisticians, chemists and biologists – to mean a well-defined function that quantifies the extent and scope of such differentials, in the spirit of Shannon information theory for uncertainty [Soofi 2000]. Two disciplines, computer science and molecular biology, have been very successful by embracing information-theoretic notions. This has led not only to the generation of vast amounts of data and information as the fields have advanced, but also to a new understanding of the concept of information itself. From this brief discussion it is clear that information as a polysemantic and flexible concept invites, and often requires, a pluralistic interpretation rather than the straitjacket of a monistic interpretation. The reason is that, emerging as it does from a social shadow, it is the basis of all communications: it helps us categorise the environment and react and cope with it, as a map, model and approach. It can represent and convey a figment of reality, and can be used to construct it. These two twin and distinct aspects of information, of representation and communication, are intimately connected, and one cannot exist without the other.

1.4.2 Examples

A few by no means exhaustive motivating examples can give a flavour for the kind of problems that can be approached by adopting this collective, interdisciplinary perspective, and all these problems have to do with storing, processing and exchanging information dynamically and reducing the dimensionality of a very complex coordination (i.e. information exchange) problem. In economics, *markets* and *exchanges* for material goods and services have existed for millennia in forms ranging from simple barter to billions of users. One need only think of street markets, supply chains, and today's online marketplaces and financial markets that are on a local, regional, international and planet-wide scale.

Another mechanism, natural (human) language, is a form of *consensus* among its users to express and communicate concepts, ideas, categories and actions. Meetings, conferences, books and signs are just the kind of problems solved by language; where they happen in sufficient temporal and spatial proximity – i.e. they become a conversation – a dynamic structure of themes and topics and turn-taking arises between its users.

The human brain is perhaps the most complex computational structure ever studied. It is composed of parts called *neurons* that continuously orchestrate a rhythmical flurry of local and global activity which is both of a conscious and unconscious nature, and likewise, despite a century of scientific advances, it is still largely unknown [Baum 2003, Pulvermuller 2002, Buzsaki 2006]. One long-standing hypothesis, now supported by physiological evidence, is that such activity is organised in groups of self-organised neurons, or *cell assemblies* that become activated and enable non-linear computation and long-term memory [Hebb 1949, Palm 1981]. Understanding information processing in such structures is one of the goals of Artificial Intelligence [Norvig & Russell 1999].

Another ubiquitous example of agoric computation is given by *money*. It has many meanings and varying worth, and a widespread economic definition of money is given by its functions as a unit of account, medium of exchange, and store of value. Yet, when one reflects about the emergence of money at the dawn of society and its historical and technological development, its most powerful function is that of being a *currency of trust*, by means of which some information about stability and trustworthiness is quantified, encoded, and memorised to facilitate exchange; agents no longer need to barter things or actions and be subject to the double coincidence of wants.

1.5 Research Questions

The main research aims guiding our thesis can be summarised in the following question: can we translate the unifying perspective of agoric systems into computational objects and models that have relevance for solving everyday problems, perhaps by simulating them as multi-agent systems before deploying them into production? As part of this wider question, we also seek to address a few, more refined, objectives for our research, namely

- ▷ Can we build software for such systems that is scalable and extensible?
- ▷ How do we move from raw data sensed from the environment to higher levels of aggregation (such as trust)?
- ▷ Can we apply our approach to concrete CPS and IoT applications?

and with respect to these aims we have endeavoured to give answers in their proper context. There are many mathematical frameworks and algorithms that revolve around iterating a form of information propagation until it reaches a desired outcome or fixed point; we prepare the ground for this in Chapter 2. These draw on a rich variety of quantitative modelling techniques that can be used to model information flow in multi-agent systems and the basic idea of a dynamical system as a composition[2] of smaller units that are updated – and possibly created and destroyed – dynamically as computation proceeds, where the flow can be from inputs (such as percepts, observations) to outputs (actions, decisions) and incorporates feedback from both the environment and the agent's goals. In principle, studying this kind of agoric computation is straightforward, and gives scope for a common perspective.

In practice, despite their amenability to rule-based behaviour or procedures at various levels of abstractions in a social setting, a major shortcoming of the current state of the art in agoric systems surveyed in the literature on open, autonomous systems is given by the dearth of practical computation models that can orchestrate the relationships between data and physical components, including controllers, sensors and actuators, and in particular there is a distinct lack of software models that bridge streaming sensor data into higher level abstractions, extracting features and performing actionable decisions on systems that are supposed to continually run, evolve and adapt to environmental changes. That is, doing the 'synthesis' and 'data feedback' steps of the computation and doing it at scale in a noisy environment remains a challenge for software engineering, as reflected in the scant work done on software implementations from these applications.

This becomes apparent in the lack of a truly computationally grounded model for trust, which is of great importance in real-life systems. Trust is a quintessential example of a forward-looking belief or state of expectation (positive or negative) towards other agents' actions and intentions of not exploiting the truster's vulnerability, and is intimately connected with how to model new situations and changes in the environment; in contrast with reputation, one does not look back at some past attribute or experience

[2] i.e., as a choice of interfaces, assembling/composition, and nesting.

but at at a future, ongoing higher level description that facilitates the functioning of a system, and thus trust provides a natural testbed for a greater understanding of agoric systems.

As these introductory considerations stress, the flow of information and its operational use in agoric systems is their most defining characteristic. To this effect, we dwell for a large part of the thesis on software implementations where we can apply the "glue of trust" to hold a system together. In particular, there is no obvious reason by which one should not model trust as a higher level function stemming from the cost of remembering information balanced with future opportunity, which relies on determinants such as (a) the response of an agent – or group of agents – to other agents' trustworthiness, and vice versa, evaluated in response to new, possibly contradictory information; and (b) the willingness of agents to accept risk and vulnerability arising because of partial information, possibly evaluated as thresholds.

1.6 Contribution and Thesis Overview

In our work, we have adopted a theoretical perspective for the implementation of agoric computation – i.e. calculi for trust, and graph based models for information networks; an engineering perspective, via software artefacts that implement our theory programmed in Java, Jason, Brahms and NetLogo; and provided data analysis for the validation of our proposal. This has involved implementing a streamlined, two-stage approach for information processing, based on specifying the topology (or interaction model) and the update rules (or information exchange model), in line with the discussion of agoric computation of Subsection 1.4.1.

The work in this thesis has benefited from extensive discussions and interactions with fellow researchers towards refining the theory and solving practical problems. Some of the ideas discussed here have led to substantial collaborative work and to the implementation of code that resulted in the following publications in conferences and journals:

- ▷ [1] Mirco Bordoni, Michele Bottone, Bob Fields, Nikos Gorogiannis, Michael Margolis, Giuseppe Primiero and Franco Raimondi (2015) *Towards cyber-physical systems as services: the ASIP protocol.* In: 2015 IEEE/ACM 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS), 17-17 May 2015, Florence, Italy.

- ▷ [2] Michele Bottone, Giuseppe Primiero, Franco Raimondi and Vincenzo De Florio (2016) *A model for trustworthy orchestration in the Internet of Things.* In: 2016 12th International Conference on Intelligent Environments (IE), 14-16 Sept 2016, London, United Kingdom.

- ▷ [3] Michele Bottone, Giuseppe Primiero, Franco Raimondi, and Neha S Rungta (2016) *From raw data to agent perceptions for simulation, verification, and monitoring.* In: 12th International Conference on Intelligent Environment 2016:- 5th International Workshop on Reliability of Intelligent Environments (WoRIE'16), 14-16 Sept 2016, London, United Kingdom.

- ▷ [4] Michele Bottone, Filippo Palumbo, Giuseppe Primiero, Franco Raimondi and Richard Stocker (2016) *Implementing virtual pheromones in BDI robots using MQTT and Jason.* In: 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), 03-05 Oct 2016, Pisa, Italy.

- ▷ [5] Giuseppe Primiero, Michele Bottone, Franco Raimondi, and Jacopo Tagliabue (2016) *Contradictory information flow in networks with trust and distrust.* In: 5th International Workshop on Complex Networks and their Applications (COMPLEX NETWORKS 2016), 01-06 Dec 2016, Milan, Italy.

- ▷ [6] Giuseppe Primiero, Franco Raimondi, Michele Bottone and Jacopo Tagliabue (2017) Trust and distrust in contradictory information transmission. *Applied Network Science*, **2** (12), ISSN 2364-8228, Springer.

- ▷ [7] Michele Bottone, Franco Raimondi, and Giuseppe Primiero (2018) Multi-agent based simulations of block-free distributed ledgers. *In: E3WSN2018, Co-located with the 32nd IEEE International Conference on Advanced Information Networking and Applications (IEEE E3WSN AINA-2018)*, 16-18 May 2018, Pedagogical University of Cracow, Cracow, Poland.

My contribution to the publications cited above has been, as a rule, proposing and discussing the extensions of the underlying models and programming them using agent development environments, collecting and presenting data, and for four of them, writing the papers in full. In particular, this has involved specifying models for applying the glue of trust in information flows [2, 5.6], devising solutions to move from raw sensor data to higher levels of abstractions or trust [3, 4] and performing applications to trust in CPS and IoT [1, 7] with detailed evaluations.

This thesis is structured as follows: this introductory Chapter broaches the subject matter on agoric computation as formalised information flow and the research questions that form the focus of this doctoral research; Chapter 2 gives a critical review of the relevant literature related to modelling agoric open systems, in particular several mathematical formalisations and programming frameworks that approach open systems under different guises but can be shown to be manifestations of the same underlying mechanism, and dwells on trust models and cyber-physical systems; Chapter 3 introduces preliminary concepts useful for fleshing out the vision of this thesis, namely cyber-physical components such as sensors, actuators, controllers, and software abstractions such as the communication protocols and the publish/subscribe framework, the Java-based Jason and Brahms and NetLogo programming environments for multi-agent systems, the natural deduction calculus, and graph theory; Chapter 4 provides the programming models underlying four thematic areas in the work published to date as part of this thesis, in particular bridging from low-level agent percepts and data to higher-level representations of information, an operational formalisation of trustworthiness in orchestrated systems, a proof system and natural deduction calculus and associated algorithms for computing the cost of trust in several types of networks of agents seeded with contradictory information which is then transmitted along the network, and a directed graph structure for distributed ledgers; Chapter 5 presents in some detail the software implementations for the four thematic areas of the preceding Chapter; Chapter 6 gives a reasoned evaluation of the results of some of the experiments performed that have been published in workshops and conferences in the past two years, with annotations; finally, Chapter 7 takes stock of the progress achieved in this thesis and the current limitations of our approach and concludes with a roadmap and several leads for further work and future extensions.

Chapter 2

Related Work

In Chapter 1 we gave an introductory treatment of the main focus of this thesis, namely how to model the propagation of information and the exchange of relevant data in open systems of agents. While the subject of static topologies and environment has received considerable attention in the wider literature of several disciplines, that of information flow in dynamic environments has received far less attention, and software engineering applications are still in their infancy. In the rest of this Chapter, we give in Section 2.1 a broad brush of related work in the areas that form the scope of our thesis, deferring the detailed discussion of logical and programming approaches to Chapter 3; we also give an overview of the wider literature on trust in multi-agent systems in Section 2.2.

2.1 Mathematical Formalisms

Approach	Directional	MAS	Composable	Scalable	Implementation
1. Probabilistic	✓	✓	✓✗	✗	✗
2. Graphical-Process	✓	✗	✓	✗	✗
3. State-Machine	✓	✗	✓	✓✗	✗
4. Influence-Systems	✓	✗	✗	✓✗	✗
5. Category-Theoretical	✓	✓	✓	✓✗	✗
6. Game-Theoretical	✓✗	✓	✗	✗	✗
7. Propagators	✓	✓	✓	✗	✗

Table 2.1: Current Formalisms for Agoric Computation

There currently exists a plethora of modelling and analysis approaches for agoric open systems, which have led to thousands of new papers in the past decade alone. The summary classification given in Table 2.1 is not intended to be exhaustive as, appropriately enough, modelling approaches are interlinked and straddle these boundaries; however, it provides a quick synthesis and comparison of the currently existing formalisms with respects to the gaps identified in Section 1.5, which anticipates the critical evaluation of Subsections 2.1.1–2.1.7. In particular, we consider the following shorthand: *Directional* probes whether the given approach provides a formalisation of directed information flows; *MAS* considers whether it is directly transposable to the language and techniques of multi-agent systems, especially the BDI logics; *Composable* if the formalism allows a choice of interfaces, assembling/dissembling, composition and nesting into higher hierarchies; *Scalable* if the approach works equally well for a few components building to a very large but finite set of components; and *Implementation* if there exists a working software engineering implementation and deployment of the previous characteristics. The symbols ✓and ✗ mean that the corresponding feature is addressed – respectively, not addressed – by the relevant formalism; we use ✓✗ to highlight that is is only partially addressed. From the above considerations, we anticipate that there remain considerable gaps in the existing approach to formalisation which motivate the work presented in this thesis.

2.1.1 Probabilistic Models

The mathematical study of the "flow of information through networks" has traditionally been the subject of the field known as *Interacting Particle Systems*, under the lasting influence of [Liggett 1985, Liggett 1999]. This covers such diverse subfields as statistical physics, epidemiology, broadcast algorithms on graphs, ad hoc sensor networks, and social learning theory (such as the transmission of gossip and so-called memes over a social network). In these models, the specification of either the meeting model (the geometry or topology) or update rule (the dynamics) is a probabilistic one, where the next step occurs at random according to a specified distribution. In a recent series of papers, the probabilist David Aldous and his coauthors have looked at some of these rules under the name *Finite Markov Information Exchange Processes* [Aldous 2013, Aldous & Lanoue 2012, Aldous *et al.* 2014]; see also [Sainudiin & Welch 2015]. These models, broadly speaking, start from the structure of a population of individuals with a well-defined topology and study the state of the individuals as some token – which may be simply the description of its state, as in epidemic models – is exchanged, and some property of the network is established. [Louzada Pinto 2016] gives an extensive overview of probabilistic models for information diffusion and opinion dynamics in social networks. These approaches are well suited to formalising the directionality of information flows in a MAS setting and, depending on the meeting model, can address levels of hierarchies although they are only partialy composable; from the viewpoint of this thesis, purely probabilistic models are not scalable and there exist no real-world implementations.

2.1.2 Graphical Models and Process Calculi

Ambient Calculus is a process algebra focused on mobility, introduced in [Cardelli & Gordon 1998, Cardelli & Gordon 2000] as a single framework to describe mobile agents, the environment where agents interact and the mobility of environments themselves. As such, it encompasses both mobile computing, concerning computation that is carried out in mobile devices, and mobile computation, which is code that moves between devices, such as apps or software agents. An *ambient* is thus an abstract entity that can be used to model different elements in physical space (i.e. agents and locations) and cyber space (i.e. programming scopes and variables and interactions). Ambients inhabit a hierarchy of locations and create a tree structure that can be dynamically reconfigured when agents or their ambient perform a set of capabilities or actions. A formula in the ambient calculus may be both a representation of the structure of an environment – how ambients are structured and nested – and its evolution – how its structure changes through the execution of a given set of capabilities. The ambient calculus formalises location effectively, but performs less well when describing the real-world, complex dynamical interactions of a large number of elements, such as security intrusions [Jansen 2002]. *Bigraphs* [Milner 2009] were introduced by Milner and coauthors as a modelling formalism and meta-calculus for unifying and understanding the complexity of seemingly disparate structures in ubiquitous computing and calm technology [Weiser & Brown 1996]. The main idea of bigraphs is to consider both communication and nesting within the same framework; to this effect a bigraph consists, as the name implies, of two graphs: a *place graph* – which can be a tree or even a forest – that captures notions of locality or containment, and a *link graph* that models the communication or association between modules or units that may represent agents and environments. The place graph consists of a set of nodes, each of which is given a control drawn from the bigraph signature, i.e. a unique name and an arity that determines the number of connecting ports that are available at a given node to be interconnected with edges in the link graph (controls can also be passive if they engender no reaction, and because links live in a hypergraph, multiple ports may connect to a single name). Bigraphs can be studied both visually, and by prefixing terms or their names. *Bigraphical reactive systems (BRS)* extend the bigraph syntax with a rigorously-defined semantics of dynamic behaviour, expressed as a set of *reaction rules* of the form $R \to R'$, where R is a *redex* and R' is a *reactum* with which the matched portion of the bigraph should be replaced, i.e. if this portion matches the rule pre-condition, it is rewritten as stated in the post condition. Reaction rules express the possible actions that agents may perform, and allow reasoning about the evolution of topological configurations of the environment. Both ambient calculi and bigraphical reactive systems generalise earlier formalisations for communicating agents, such as the π-calculus [Milner *et al.* 1992, Riely & Hennessy 1998] and the join-calculus [Fournet & Gonthier 1996, Fournet *et al.* 1996]. Despite their being well-suited with respect to directionality flows and composability, graphical process calculi are ill-suited to MAS environments and also find difficulty in achieving scalability and ease of implementation.

2.1.3 State-Machine Models

State-machine based models try to extend the algorithmic, functional, Turing-Machine based model of computation into a more expressive architecture suitable for modelling interaction of object-oriented and more generally distributed systems of computing agents, which are large but finite. A formal model of interactive computation using *se-*

quential interaction machines (SIM) was suggested by Wegner [Wegner & Goldin 1999] as a substrate for computation which is potentially more powerful than non-interacting algorithms. Informally, SIMs are stream-processing machines that model sequential interaction by input-output streams, where each computation step may be viewed as a complete Turing Machine computation – i.e. the working out of a computable function – and introduces dynamic dependence of inputs on prior outputs. A related dynamic state-machine based mathematical framework allowing the modelling of individual components and their interactions and changes was recently proposed by [Attie & Lynch 2016]. *Dynamic Input-Output Automata (DIOA)*, based on I/O automata [Lynch & Tuttle 1989, Lynch 1996], are state-based machines that perform three kinds of actions: *input*, *output* and *internal*, and produce an externally visible *trace* or set of traces by removing all internal actions and replacing each state by its external signature – i.e. its input and output actions – and replacing blocks of identical external signatures by a *representation*. Unlike I/O automata, which are static, DIOA model a dynamical system as an automaton that is *one level higher* in the hierarchy and can be repeated and nested, allowing the creation, modification and distruction of entire subsystems with a single action. Thus DIOA have the ability to dynamically change their signatures (i.e. the set of actions in which the automaton can participate) and create other I/O automata representing local and global configurations and to express changing capabilities. DIOA also feature parallel composition, action hiding, and action renaming operators. Much like graphical process calculi, SIMs and DIOAs suffer from being unsuitable for the implementation of MAS environment, despite good theoretical properties and partial scalability.

2.1.4 Influence Systems

Influence systems were proposed by [Chazelle 2012] as a level of abstraction based on the language of the natural algorithms of Lindermayer type, or L-systems [Prusinkiewicz *et al.* 1996]. An *influence system* is specified by two sets of functions, an action function f and communication function $\mathcal{G}$, the latter modelling the flow of information in the influence network of the agents. It is a discrete-time dynamical system $\mathbf{x} \mapsto f(\mathbf{x})$ in $(\mathbb{R}^d)^n$, where n is the number of agents and d the dimension of the ambient space, and each coordinate x_i is a d-tuple encoding the *location* of an agent i. To any state $\mathbf{x}$ one associates a directed[1]communication graph $\mathcal{G}(\mathbf{x})$ with one node per agent, and each coordinate function f_i on the map takes as input the neighbours of agent i in $\mathcal{G}$ and their location, and outputs the new location $f_i(\mathbf{x})$ of agent i in $(\mathbb{R}^d)$. By separating f and $\mathcal{G}$ the model enables the separation of the syntax (who talks to whom) from the semantics (who does what), and both f and $\mathcal{G}$ can be arbitrary functions that can be evaluated by deterministic or randomised algorithms. Simple influence systems have

[1]An undirected communication graph creates a bidirectional system.

been shown to explain several emergent phenomena and a subclass of these, diffusive (i.e. if f keeps each agent within the convex hull of its neighbours) influence systems, always converge to a consensus in a finite time. Influence systems formalise information flows well and can partially scale, but perform poorly when viewed through the lens of MAS logics, composability, and implementation.

2.1.5 Category-Theoretical Models

Informally, a *category* [Mac Lane 1998, Awodey 2006] consists of mathematical objects, morphisms or functions between objects, and structure-preserving maps between objects. An *operad* is similar to a category, with the difference that instead of a homotopy set $\hom(\mathcal{X}, \mathcal{Y})$ of morphisms from any object $\mathcal{X}$ to any object $\mathcal{Y}$, one has a set $\hom(x_1, \ldots, x_n; y)$ of operations from any finite list of labelled objects $x_1, \ldots, x_n$ to any object $\mathcal{Y}$. For an operation $f \in \hom(x_1, \ldots, x_n; y)$ one calls the $(x_1, \ldots, x_n)$ the *inputs* of f and y the *output* of f. Operations can be nested and composed in an operad, creating new operations; there also exist unary operations $\mathbf{1}_x \in hom(x; x)$ serving as identity mappings of compositions, and an associative law that makes the composite of composites well-defined. Operads are similar to symmetric monoidal coloured categories [May 1972, Leinster 2004] and enable one to define an algebra on them, that is a homeomorphism that turns the abstract operations in the operad into operations on finite sets. Thus operads formalise the intuitive notion of a structure that is constrained to have some form of consistency or self-similarity. One interesting structure is the operad of wiring diagrams [Vagner *et al.* 2015, Lerman & Spivak 2017, Spivak & Tan 2015, Rupel & Spivak 2014, Spivak 2016, Yau 2017], studied by [Rupel & Spivak 2014, Spivak 2016, Lerman 2017, DeVille & Lerman 2011] as a formal model for agent interaction. Much like on a computer logic board, a *wiring diagram* is a fixed, network-like arrangement of nodes and directed edges where each node is a box with differentiated input and output ports and one can "zoom in or out" by cutting-and-pasting graphs into other nodes. The basic idea of these diagrams is that if one inserts a wiring diagram into each of its interior boxes, one obtains a new wiring diagram and can study a complex arrangement at any level of granularity, via their hierarchical, modular structure. These structures are in a sense, universal; for example the entire web of financial relations connecting banks, firms, governments, and other economic agents can be described using the simple transcription rules of double entry accounting [Katis *et al.* 2008], one of the oldest forms of bookkeeping. Category theoretical models have the best fit of the possible frameworks, allowing for directionality, MAS, composability and partial scalability. However, the assumptions and details required for consistency of the framework make them ill-suited for implementation through nimble computational objects.

2.1.6 Game-Theoretical Models

The *theory of games* [Fudenberg & Tirole 1991, Myerson 1991] concerns the mathematical study of rational, strategic interactions between individuals and groups, rather than ludic pursuits; i.e. agent behaviour in any situation where each agent's optimal choice may depend on a forecast of other agents' responses. Such agents can act exclusively out of their self interests, or cooperate towards a common goal, and develop strategies as sequences of actions. Game theoretic approaches have been useful in many fields, including computer science [Nisan *et al.* 2007]. The basic analytical structure of a game - that of maximising some payoff or reward and/or minimising some cost or penalty and looking for decision equilibria, or fixed points where actions do not change - allows for a rich choice of applications, including static and dynamic games, with completely observable and incomplete information, deterministic and stochastic, which we do not review here. From the viewpoint of this thesis, a model for a game can be used to establish whether some computational property holds; for example, game semantics underpins programming languages and computation in a syntax-independent manner [Abramsky 1997, Yamada 2017, Yamada & Abramsky 2017] and stochastic games have been used to verify probabilistically (via the model checker PRISM) the correct reasoning and functioning of autonomous systems [Kwiatkowska 2013, Chen *et al.* 2013]. Finally, game theory can help design better multi-agent systems [Parkes & Wellman 2015] and formalise lower-level representations of the data as local solutions to optimisation problems that can then be fed into higher-level features representing more abstract aspect of the data gleaned from the environment, thus providing a semantics for the flow of first-order information in deep learning [Balduzzi 2014, Balduzzi 2016]. Along with influence systems, game-theoretical approaches have been the worst-scoring among those reviewed here, only fully compliant with a partially directional MAS modelling environment and have encountered difficulties in being amenable to composability, scalability and implementation.

2.1.7 Propagator Models

The architecture of *propagator networks* was introduced by [Radul 2009, Radul & Sussman 2010] as an expressive substrate for computation and as a general-purpose distributed programming model, suitable for developing programs that are robust, nimble, adaptable, and capable of dealing with fundamentally noisy, uncertain and contradictory environments. It gives an interesting and powerful formalism for constructing decomposable software agents.

The ingredients of a propagator network are *cells* and *propagators*. The cells' job is to remember things and thus permanently store state; conversely, the propagators' job is to compute and update the cells they are connected to. Compared to most programming paradigms [Abelson *et al.* 1996], propagators are a bit like procedures in a language, and cells are like memory locations. A big difference is that cells *cumulate partial information* – which may involve arbitrary internal computations – and can therefore have many propagators reading information from them and writing information to them in a manner that is reminiscent of *publish-subscribe patterns* [Birman & Joseph 1987].

To visualise a propagator[2] architecture, it helps to consider it as a special type of wiring diagram; i.e. a system consists of computational boxes connected by wires. Each box – the propagator – watches its ports – the cells it is connected to – and reacts to the signals it understands, in turn transmitting new signals through other ports[3].

Networks of propagators have no constraints on their topology, such as acyclicness, and the flows through boxes and wires are inherently bidirectional: in principle, information can arrive on any port and it can propagate to any port. The ordering of operations in propagator networks is undefined outside of the implicit ordering caused by the structure of the propagator network itself (i.e. that a cell must update before a propagator attached to it may fire). This, along with the strong separation of state from computation makes data propagation a flexible framework for concurrent programs. Partial propagator networks may be abstracted and reused as "compound propagators" by composition. As these compound propagators become active, they construct the partial propagator network represented by the compound propagator, permitting more complex operations, recursion and the construction of loops. This message-passing view of information flowing through propagator networks that compute in real time is the key characteristic of this programming model. The information stored in a cell may be updated at any time by a propagator which sends a message with new information to that cell, and as a result, updates it. The contents of that message are merged with the data currently in the cell using an appropriate, iterative merge operation. After a cell has changed its state by merging an update, any propagators that have registered an interest in the cell wake up and begin to process, using the new data as needed. These propagators may then send updates to other cells and cause another cycle of cell merging and notification of propagators; this drives continued computation. It is important to note that each cell maintains, in a way, not "a value", but "all the information it has about a value", which could be nothing (nil, "I don't know anything", or some concept of "no idea"), everything (something known with certainty: for example, a number or interval), some logical proposition or belief, or a probability or even *contradictory information* which must be resolved. Handling of contradictions forms part of the software implementation of the network, rather than the underlying propagator network, via the design of specific data structures, for example *truth maintenance systems (TMS)* [Radul 2009, Radul & Sussman 2010] that have features

[2]Which, we remind, is a function that takes values based on data in zero, one, or more cells, and may store output in one or more output cells.

[3]Each box ignores inputs it does not know how to handle, and combines evidence from multiple ports to reach conclusions about the values that should be propagated to other ports.

which make them appropriate for handling both partial information and contradiction resolution in their merge operation. Propagation networks can also be used to incorporate logical abstractions, with the objective of introducing flexible planning. For this, a form of knowledge representation and logical connectives, propositional and deductive calculi, as well as modal and temporal logics and model checking must be introduced. A proposition in an epistemic construction can have beliefs attached and expressed in terms of orthogonal axes of belief [Jacobi 2013]: *acceptance, rejection, contradiction, knowledge*, and *ignorance*[4], treating goals, knowledge and perceptions as independent beliefs which may be connected by a network that propagates beliefs, so that each state is only loosely connected to the others. Individual beliefs are mapped to cells, and the *rules* that relate beliefs may be programmed as propagators which connect belief cells and combine belief states to generate another. This propositional system is similar to that of BDI logics, and as such supports the finding that propagators possess many of the features required for modelling and engineering agoric systems, such as directionality, composability; scalability and practical implementations have however not been established.

2.2 Computational Trust and Cognitive Agents

Trust is a quintessentially social feature, and invites reasoning in a multi-agent framework with the added complication of reasoning about delegation and transitivity, i.e. who trusts whom, and whether third parties with social connections in common are also trusted. Conversely, reasoning about trust naturally leads to questions about trustworthiness in a system of agents, that is the expectation that someone/something can be trusted, and to questions about trusted information and data provenance in contexts where agents might have to rely on external sources to execute decisions effectively and securely. Although trust as studied in the literature is mainly a human endeavour, the subjects and objects of trust are not just limited to human agents, but can be extended to and from computational devices and machines that can act autonomously.

An overview of trust in open systems and related approaches appears in the recent collections [Reif *et al.* 2016] and [Abbass *et al.* 2018]; see also [Abbass *et al.* 2016] for its application to data-rich autonomous systems. One aspect of distributed computational thinking that is especially challenging is that of trust and its assessment, i.e. the quality of being trustworthy. Trust learning is increasingly crucial in information exchange, negotiation, and any other kind of social interaction amongst autonomous agents in open systems; yet most current models for computational trust lack the ability to take data and context into account when trying to predict future behaviour of a system of interacting agents.

[4]Acceptance and rejection correspond to belief in the truth and falsehood of the proposition, respectively.

One active topic of research in the domain of MAS is that of computational trust models. Of the many social structures, norms and conventions that arise out of the relationships of agents who engage in repeated interaction, several are borne out of the attitudes or inclinations that individuals have towards each other. *Trust* [Gambetta 1990, Bühlmann 1979] is one such mechanism for managing the uncertainty about these autonomous components and the information they deal with. Closely related fiduciary concepts such as distrust, mistrust, trustworthiness and confidence are quintessential characteristics of human and artificial societies; they are required for successful transactions and exchange, and at the same time they play an important role in reducing the complexity of coordination, cooperation, delegation and decision-making. While a defining feature, the inherently social nature of trust [Gambetta 2001] also makes it particularly challenging to study computationally.

Like agoras and information, these trust structures, norms and conventions are so pervasive that it could be argued that they provide the foundation for any operational notion of agency, and the breadth of situations where they can be applied is testament to their explanatory power. At the same time they are notoriously difficult to pin down computationally, pertaining to rationality and emotion, direct and indirect evidence, empirical validation and reputation, morality and social standing, intransigence and power, bias and pragmatism.

Trust is an an emergent property of multi-agent systems: it exists as a sequence of decisions in time towards well-defined beliefs or actions, and it creates expectations into the future. It often takes a long time to build trust, it can be self-reinforcing, and it can quickly evaporate. It is also costly in that the trusting agent is exposed to risk, i.e. the trusted agent, or trustee, may fail to execute some action or contract that it is capable of fulfilling.

Research into trust-related models is a very active topic in artificial intelligence and multi-agent systems [Pinyol & Sabater-Mir 2013], based on foundations from many different disciplines: sociology [McKnight & Chervany 2001], cognitive science [Castelfranchi & Falcone 2010], argumentation [Besnard & Hunter 2008], statistical science [Teacy *et al.* 2006, Matt *et al.* 2010], game theory [Burnett *et al.* 2011, Huang & Kwiatkowska 2017], network science [Golbeck 2006, Mui 2002, Tsang & Larson 2014] to name a few.

In a computational trust model, one defines possibly multiple methods for calculating a truster agent's trust in a trustee based on the agent's own interactions with the trustee, as well as on information that is available in the environment about the trustee. The trust model then aggregates this information and evaluates it for a given trustee or for the system as a whole.

Several influential theoretical models for computational trust have been advanced before. One popular approach is based on *quantified direct interaction* [Marsh 1994] and its extensions, which defines trust as a numerical value that arises out of three different contextual notions, namely basic trust, general trust and situational trust. Relevant information may be direct communications from other agents in the system, sharing their own trust evaluations of the trustee, reputation, indirect experience, or any other source, within each of these contexts. The second influential trust theory is based on the *socio-cognitive approach* [Castelfranchi & Falcone 2001, Castelfranchi & Falcone 2010], that contends that only a cognitive agent – i.e. one endowed with beliefs, goals, mental attitudes, intentions – can trust another agent, because of the dynamic interaction of preference, knowledge, expectation and new information, leading to a decision whether to trust or not. In this interpretation, trust depends on the agent's own general beliefs and possibly prior opinion about the capability of the source to either (a) convey some useful information; (b) perform some dependable action. If an agent needs to trust a trustee, it is because it needs something from the trustee that could help it fulfill its own goals. This capability can take, for instance, the form of shared goals or coincidence of wants. Cognitive social trust then arises as an offshoot of cognitive agents who reason for and against adopting beliefs, committing to goals and pursuing actions, and in particular the mental states of the trustee relating to actions in the future that might affect the truster. Of course in the real world, information can be incomplete; accommodating this, and the dynamic of trust, is particularly challenging when one considers that, in most of the social trust models in the literature, an agent's computational trust model does not change when the agent senses a change in the environment. Within the cognitive framework, [Herzig *et al.* 2009, Herzig *et al.* 2008, Koster *et al.* 2013] have formalised and studied modal epistemic logics based on BDI logics [Rao & Georgeff 1998b]. [Da Costa Pereira *et al.* 2015, Paglieri *et al.* 2014] suggested argumentation as a mechanism for constructing reasons for trusting or distrusting other agents, while a dependence-based modal logic is explored in [Patti 2002, Singh 2011, Kramdi 2015]. [Parsons *et al.* 2012] have provided a taxonomy for argument schemes when reasoning about trust, and [Koster *et al.* 2013] study trust from the agent's perspective as a black box with the various information sources as input and a trustworthiness evaluation of the trustee as output. Information propagation in trust problems has been studied in [Guha *et al.* 2004, Ziegler & Laursen 2005, Quercia *et al.* 2007, Hang *et al.* 2009, DuBois *et al.* 2011]. Other researchers have been combining an argumentative stance with evidential statistics [Matt *et al.* 2010] formulating trust in the context of the subjective probabilistic definition [Jøsang *et al.* 2006], and using either Bayesian approaches [Lahijanian & Kwiatkowska 2016], and stochastic multiplayer games [Huang & Kwiatkowska 2017]. Trust-based interaction in mobile computing has been investigated by [Setter *et al.* 2016], while the sudden trust collapse in networked societies has been addressed in [Da Gama Batista *et al.* 2015].

Chapter 3

Background

In this Chapter we introduce the preliminaries and commonalities in this thesis. In particular, we give a definition of CPS and the publish-subscribe protocol MQTT in Sections 3.1 and 3.2, introduce various software tools used in this thesis, such as Jason (Section 3.3), Brahms (Section 3.4) and NetLogo (Section 3.5), and the logical (Section 3.6) and mathematical (Section 3.7) underpinnings of our work. The following toolbox summarises the background of tools and contexts used in the remainder.

Toolbox

1. **Theory**

   ▷ **Natural Deduction:** an inference system used to construct logical proofs in steps

   ▷ **Graph Theory:** the natural abstraction for networks and other relationships

2. **Engineering Solutions**

   ▷ **Jason/AgentSpeak:** a rule-based BDI agent programming language and environment

   ▷ **Brahms:** an alternative, activity based programming environment

   ▷ **NetLogo:** an interactive agent modelling and simulation environment

3. **Validation**

   ▷ **Cyber-Physical Systems:** orchestrations of micro-controllers, sensors and actuators

   ▷ **MQTT:** a lightweight communications protocol for publish/subscribe patterns

3.1 Microcontrollers, Sensors and Actuators

CPS are typically built using multiple interacting components at varying levels of abstraction, modularity and connectivity. These usually serve a specific purpose in the system, whether interfacing with the external environment, reading or writing data to it, or processing and making sense of streams of data.

- ▷ **Sensors** are components that accept and read data from the environment, and can thus be thought of as *input ports* of a tightly coupled, modular, composite element. Depending on its overall design, sensors in a cyber-physical system can be set up and defined for a range of parameters (temperature, acceleration, pressure, light, gyroscope, humidity, depth of field, etc) and sophistication, accuracy and/or sensitivity.

- ▷ **Actuators** are, conversely, components that effect and write data to the external environment, and perform some action on behalf of the system. Typical examples are servo motors, relays, displays, stepper motors, illuminators, hydraulic arms, launchers, conveyors and so on. They can be connected to other parts or components, justifying their classification as *output ports*.

- ▷ **Controllers** are the glue that binds sensors and actuators together. These components usually need to be programmed individually to take into account both low-level implementation details and the high-level requirements of the application – or *function* – of which the controllers are part. As the name implies, their purpose is to control the flow of data and the logical computation in the system. Modern controllers are highly modular, finely grained and miniaturised; even application processors on a modern mobile platform such as a smartwatch, phone, tablet or drone are often complex systems-on-a-chip. Specialised systems many rely on a single microcontroller; most applications require several microcontrollers interfacing and acting in concert. In the past decade, the Arduino microcontroller platform has become hugely popular [Margolis 2011].

3.2 MQTT and Publish/Subscribe Protocols

The *(topic-based) publish-subscribe pattern* [Birman & Joseph 1987] has emerged as the underlying communication mechanism of choice for a wide variety of CPS applications. In this pattern, entities that create messages (the publishers) are not aware of the potential receivers. They instead "broadcast" messages, whereas the receivers, on the other hand, subscribe to messages; both the broadcast and/or subscription may be performed through a broker. In topic-based publish-subscribe architectures each message has a *topic* and subscribers only receive messages for the specific topic they are interested in. Accordingly, publishers need to provide a topic for each message that is generated.

There are a number of open protocols for messages using the publish-subscribe communication model that are meant to provide a lightweight, asynchronous alternative to HTTP/REST in the context of Cloud Computing and the Internet of Things for coordinating many separate environments with minimal configuration and overhead, among them MQTT, XMPP, and AMQP[1].

Message Queue Telemetry Transport (MQTT) is a lightweight protocol initially developed for wireless sensor networks [Hunkeler *et al.* 2008] and running on top of TCP/IP connections. MQTT is very efficient and simple to implement, and was specifically designed for high-latency, resource-constrained devices with low bandwidth and power draw, features that make it ideal for use in embedded systems. MQTT messages are characterised by a *topic* and by a *Quality of Service* (QoS). A *broker* is required to dispatch messages from publishers to subscribers. When a client connects to the broker, it is identified by an ID and it can perform the following actions: connect, disconnect, subscribe (to a topic), unsubscribe (from a topic) and publish a message under a certain topic and at a given QoS. Topics are organised in a hierarchy: for instance, the message `t1/t2/t3/t4` has `t1` as main topic, `t2` as subtopic, `t3` as subsubtopic, and so on. Subscribers can specify *patterns* using the symbols + (matching one occurrence of a topic) and # (matching any number of topics). For instance, `t1/#/t4` will match `t1/t2/t3/t4`, but `t1/+/t4` will not.

MQTT provides three levels of Quality of Service:

- Level 0: the message is sent *at most* once (either by the client or by the broker), with no guarantee of delivery.
- Level 1: the message is sent *at least* once, until a confirmation is received.
- Level 2: the message is sent *exactly* once.

Several open source implementations are available, both for brokers and for clients. In our experiments we have employed the on-line broker HiveMQ[2] and the associated on-line MQTT client[3]. The latter allows both subscription and publishing of messages. A local broker can also be implemented using the Python-based tool Mosquitto[4].

3.3 Jason

Jason is essentially a *rule-based system* that makes use of the notion of planning implicit in BDI logics [Bratmann 1999]. Its underlying structure is the concept of a *reasoning cycle* we mentioned in Subsection 1.2.1, coupled with the notions of beliefs, desires, intentions and plans. The software architecture of Jason is based on an extension of the agent programming language *AgentSpeak* [Bordini *et al.* 2007] which is an abstract,

[1]The latter two have support for extensions to more complex scenarios, APIs and domains.
[2]`http://broker.hivemq.com`
[3]`http://www.hivemq.com/demos/websocket-client/`
[4]`http://mosquitto.org/`

declarative programming language for implementing BDI agents with Prolog-like instructions, that can be extended to fit specific needs. Its syntax defines agent programs as a set of logical beliefs, rules and plans, and is formally defined in the following way.

For $\mathcal{S}$ a finite set of symbols including predicates, actions, and constants, and $\mathcal{V}$ a set of variables, one can define vectors of terms in first-order logic:

- If b is a predicate symbol and $\mathbf{t}$ a term, we define $b(\mathbf{t})$ to be a belief atom.
- if $b_A(\mathbf{t})$ and $b_B(\mathbf{t})$ are belief atoms, where A and B can be conjunctions, disjunctions or negations of belief literals, then the rule $b_A(\mathbf{t}) :- b_B(\mathbf{t})$ describes how the latter is inferred from the former.
- If $g(\mathbf{t})$ is a belief atom, then $!g(\mathbf{t})$ and $?g(\mathbf{t})$ are goals, $!g(\mathbf{t})$ denoting an achievement goal and $?g(\mathbf{t})$ a test goal.
- If $p(\mathbf{t})$ is a belief atom or a goal, then $+p(\mathbf{t})$ and $-p(\mathbf{t})$ are triggering events with $+$ and $-$ denoting respectively the addition and deletion of a belief to be held or goal to be achieved.
- If a is an action symbol and $\mathbf{t}$ a term, then $a(\mathbf{t})$ is an action.
- If e is a triggering event, $c_1, \ldots, c_m$ are beliefs and $q_1, \ldots, q_n$ are goals or actions, the rule $e : c_1, \ldots, c_m \leftarrow q_1, \ldots, q_n$ defines a plan, with $c_1, \ldots, c_m$ its context and $q_1, \ldots, q_n$ its body.

Jason programming [Hübner & Bordini 2006] revolves around *plans*, which are the closest thing there is to a function or method in a declarative language. *Actions* in the body of an expression are executed in sequence as a consequence of the triggering of the plan, which can consist of belief addition and removal, requests to achieve and unachieve (sub)goals, or built-in or user-defined internal actions that change the environment or the agent's mental state over time. In Jason, ground literals are also extended by strong negation, annotations, and message passing.

Jason extends the AgentSpeak syntax into a flexible, extensible Java-based, open-source development environment and interpreter, which is easily customisable. In particular, Jason allows for the definition of bespoke *environments* extending a base Environment class in Java. We exploit this characteristic in our implementations in Section 5.1 to interact with an MQTT infrastructure.

3.4 Brahms

Brahms [Clancey *et al.* 1998] is a Java-like, BDI-based modelling language for agents, with a specific target to model human-machine interactions. As a modelling environment it is perhaps closer to the possible implementation logic of complex scenarios: the

Brahms language allows for the representation of situated activities of agents in a geographical model of the world. Situated activities are actions performed by the agent in some physical and social context for a specified period of time. The execution of actions is constrained (a) locally, by the reasoning capabilities of an agent and (b) globally, by the agents' beliefs of the external world, such as where the agent is located, the state of the world at that location and elsewhere, located artefacts, activities of other agents, and communication with other agents or artefacts. The objective of Brahms is to represent the interaction between people, off-task behaviours, multitasking, interrupted and resumed activities, informal interactions and knowledge, while being located in some environment representative of the real world.

Brahms models [Bordini *et al.* 2009] are described using a Java-like syntax that allows for inheritance. At each clock tick the Brahms simulation engine inspects the model to update the state of the world, which includes all of the agents and all of the objects in the simulated world. Agents and objects have states (factual properties) and may have capabilities to model the world (e.g., a display is modelled as beliefs, which are representations of the state of the environment). Agents and objects communicate with each other; the communications can represent speech, reading, writing, etc. and may involve devices such as telephones, radios, displays, etc. Agents and objects may act to change their own state, beliefs, or other facts about the world. Brahms can be extended using *Java activities*, which are activities declared and composited by the modeller using the Java programming language, using a syntax like in the code snippet in Figure 3.1.

3.5 NetLogo

NetLogo [Wilensky 1999] is a well-known, widely used, cross-platform modelling and simulation environment for complex systems of concurrently interacting agents written on top of the Java Virtual Machine that inherits much of its advanced concurrency and library support [Tisue & Wilensky 2004], thus making it expressive and powerful and customisable. It has extensive documentation and tutorials, and also comes with the Models Library, a large collection of pre-written simulations that can be used and modified. Freely available at at `http://ccl.northwestern.edu/netlogo/docs/`, it was originally developed by Uri Wilensky of Northwestern University and subsequently used by thousands of students, teachers and researchers worldwide. Its main attractiveness comes from its being based on a Logo [Papert 1980] dialect extended to support agents and modern programming paradigms, well-suited for rapid prototyping of complex scenarios, using "turtles" (agents), "agentsets" as collections of agents that can be customised on the fly, "patches" (the spatial coordinates on which agents sit), "links" (relationships between turtles), and "extensions" – libraries that enhance the core func-

```
 activities ::= activities [ activity ]

activity ::=
   activity −one |
   activity −two |
   complex−activity |
   super−activity

// [...]

complex−activity ::=
   setup−agent |
   create−agent |
   create−object |
   create−environment |
   communicate |
   broadcast |
   get−data |
   put−data

// [...]

super−activity ::=
   super−activity activity −name({params−one, params−two})
   {
     {... link to Java code ...}
   }
```

Figure 3.1: Brahms activity declaration

tionality of NetLogo. Of the latter, we make intensive use the `nw` extension[5] to analyse the network structure of our underlying models; this simple yet flexible construction makes it straightforward, for example, to include multiple agent strategies and define functions (as `procedures`) and study statistics of the data (`reporters`); and the built-in interface and tools, in particular the 2D and 3D views, buttons, switchers, choosers and plots provide an intuitive way to look for structure as the mixture of agent strategies change. Our modelling of information transmission over networks (Section 5.3) and the simulation environment of the growth of the Tangle (Section 5.4) are programmed entirely in NetLogo.

3.6 Natural Deduction

The natural deduction calculus [Arthur 2016] is a deductive system that allows us to construct proofs of tautologies in a sequence of steps. It consists of a set of *rules of inference* for deriving *consequences* from *premises*, by building a *proof tree* whose root – usually drawn at the bottom – is the proposition to be proved and whose leaves – drawn at the top – are the initial assumptions or axioms. Thus, for instance, logical

[5]Available in versions 5.3.1 and 6.0.2.

deduction rules such as *modus ponens*

$$\frac{\Phi \qquad \Phi \rightarrow \Psi}{\Psi}$$

mean that the propositions above the line, the premises or antecedents which we know to be true Φ and $\Phi \rightarrow \Psi$, allow us to *conclude* that Ψ is true. This simple way of expressing things can be made arbitrarily complex by specifying metavariables: Φ and Ψ need not be atomic propositions but can contain a whole family of logical statements and propositions. Likewise, by compositing rules, we can introduce intermediate levels in a complex rule. When we use an inference rule as part of a proof, the metavariables are replaced in a consistent way with the appropriate kind of logical object.

Rules in the natural deduction systems are often of two kinds, which are self-explanatory: *introduction rules* introduce the use of a logical operator, and *elimination rules* eliminate it, like $\rightarrow$ that disappears in modus ponens. A useful convention when writing and parsing proof systems is to write the name of the rule on the right hand side of rule tree, with a suffix specifying the kind of rule. Some examples of natural deduction rules are listed in the following paragraphs.

Conjunction

Conjunction ($\wedge$) has an introduction rule

$$\frac{\Phi \qquad \Psi}{\Phi \wedge \Psi} \wedge\!-intro$$

and two elimination rules:

$$\frac{\Phi \wedge \Psi}{\Phi} \wedge\!-elim(left) \qquad \frac{\Phi \wedge \Psi}{\Psi} \wedge\!-elim(right)$$

$\top$ Rule

The rule for $\top$ is called “unit” and is the simplest. Since it has no premises, this rule is an *axiom*: something that can start a proof.

$$\frac{}{\top} unit$$

Implication

To prove an implication of the form $\Phi \rightarrow \Psi$, we assume Φ, then reason under that assumption to try to derive Ψ. If we are successful, then we can conclude that $\Phi \rightarrow \Psi$. In a proof, it is always valid to introduce a new assumption Φ and give it a name, then reason under that assumption. Each distinct assumption must have a different name, for example a in the example below:

$$\frac{}{[a:\Phi]}\,assum$$

A proof is valid only if every assumption is eventually discharged in the proof tree below the assumptions. This job is done for example in the implication introduction rule, in $\Phi \rightarrow \Psi$, one discharges a prior assumption $[a : \Phi]$ (Intuitively, if Ψ can be proved under the assumption Φ, then the implication $\Phi \rightarrow \Psi$ holds without any assumptions). One writes a in the rule name to show which assumption is discharged. This rule, and modus ponens are the introduction and elimination rules for implications:

$$\begin{array}{c}[a:\Phi]\\ \vdots\\ \dfrac{\Psi}{\Phi \rightarrow \Psi}\rightarrow -intro/a\end{array}$$

$$\frac{\Phi \qquad \Phi \rightarrow \Psi}{\Psi}\rightarrow -elim$$

Disjunction

Disjunction ($\vee$), conversely, has two introduction rules

$$\frac{\Phi}{\Phi \vee \Psi}\vee -intro(left) \qquad \frac{\Psi}{\Phi \vee \Psi}\vee -intro(right)$$

and one elimination rule:

$$\frac{\Phi \vee \Psi \qquad \Phi \rightarrow \Gamma \qquad \Psi \rightarrow \Gamma}{\Gamma}\vee -elim$$

Negation

A negation $\neg\Phi$ can be considered an abbreviation of $\Phi \rightarrow \bot$, with two rules

$$\frac{\Phi \rightarrow \bot}{\neg\Phi}\neg -intro \qquad \frac{\neg\Phi}{\Phi \rightarrow \bot}\neg -elim$$

Falsity

The two rules for falsity in natural deduction

$$\begin{array}{c}[a:\neg\Phi]\\ \vdots\\ \dfrac{\bot}{\Phi}\,RAA/a\end{array}$$

$$\frac{\bot}{\Phi}EFQ$$

are known as *reductio ad absurdum (RAA)* and *ex falso quodlibet (EFQ)*. The first is the basis of proofs by contradiction, and says that if by assuming that Φ is false we can derive a contradiction, then Φ must be true. The assumption a is discharged in the application of this rule[6].

Excluded Middle

Another classical tautology that is often used as an axiom is the the law of the excluded middle, or *tertium non datur (TND)*:

$$\frac{}{\Phi \vee \neg\Phi}TND$$

Proofs

A proof of proposition Ψ in natural deduction starts from axioms and assumptions and derives Φ with all assumptions discharged. Every step in the proof is an instance of an inference rule with metavariables substituted consistently with expressions of the appropriate syntactic class. A proposition that has a complete proof in a deductive system is called a *theorem* of that system.

Soundness and Completeness

We say that a deductive system is *complete* if all true statements are theorems, i.e. have proofs in the system. Completeness gives a measure of a deductive system's power. For propositional logic and natural deduction, this means that all tautologies have natural deduction proofs. Conversely, a deductive system is called *sound* if all theorems are true. Proof-writing can, in a certain sense, be considered the process of mechanising the process of deduction, where the role of rules is to provide reasoning shortcuts. Rules are sound if there is a way to convert a proof using them into a proof using the original rules. In this case, we say that such added rules are called *admissible.*

[6]This rule is present in classical logic but not in intuitionistic (constructive) logic. In intuitionistic logic, a proposition is not considered true simply because its negation is false.

3.7 Graph Theory

Graph Theory [Bollobás 1998] studies networks [Newman 2010, Barabasi 2015] where the emphasis is on barebones relationships between components. Formally, a *graph* is a collection $\mathcal{G} = (V, E)$, where V is the set of nodes and E the set of edges. For each $v \in V$, the in-degrees and out-degrees are specified by $d_{\mathbf{in}}(v) = \sharp\{e = (c_1, c_2) \in E : c_2 = v\}$ and $d_{\mathbf{out}}(v) = \sharp\{e = (c_1, c_2) \in E : c_1 = v\}$, with $\sharp$ denoting set cardinality. An edge $e \in E$ between two nodes $v_1, v_2 \in V$ can be either *oriented*, if the link is described by a direction $v_1 \to v_2$ or $v_2 \to v_1$, or undirected, in which case the relationship is symmetrical. Additionally, a *chain* is a succession of nodes $u = v_0, v_1, \ldots, v_k = v$ such that $v_j \in \mathcal{A}(v_{j-1})\, \forall j = 1, \ldots, k$ forms a directed or undirected path between them, and we say that u *indirectly references* v.

In this framework, nodes encode agents and edges encode the existence of a one-way (in the case or directed edges) or two-way relationship between nodes. Such a relationship might be some user access or transaction approval, for oriented edges, or reciprocal information transmission among them, in the case of undirected edges. If the relationship described by e has some associated numerical characteristic, we say that the edge is *weighted*, otherwise we say that it is unweighted and all edges have the same weight in the graph.

For example, each agent $v_i \in V$ might have associated an atomic formula p representing some information (numerical sensors in the case of CPS or logical information, as in Section 3.6); in this case $v_i(p)$ and $v_j(\neg p)$ denote agents i and j knowing atomic formulas p, and $\neg p$, respectively.

The notation $v_k()$ might denote an agent k who knows nothing yet, i.e. is holding empty knowledge. Consequently, the edge case $e(v_i(p), v_j())$ denotes a transmission channel from agent i to agent j such that the former can transmit p over to the latter, and case $e(v_i(p), v_j(\neg p))$ represents an admissible edge where two agents hold contradictory information, requiring a resolution procedure; case $e(v_i(p), v_j(), v_k(\neg p))$ is shorthand for the triad where one agent v_j does not hold any knowledge yet receives contradictory information from two distinct agents, i.e. there exist two edges between three nodes.

There are various descriptions of a graph $\mathcal{G}$; one could simply enumerate its components, such as $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), (4, 5), (4, 6)\}$, or draw a diagram of connected nodes and edges, as in Figure 3.2(a).

A *Directed Acyclic Graph (DAG)* is a particular type of graph, where each edge $e \in E$ has a specific orientation and in addition there are *no cycles*, that is paths of the type $v = v_1, \ldots, v_k = v$ for any v and k. For example, acyclicness means that the tetrad of vertices $\{2, 3, 4, 5\}$ in 3.2(b) is not closed, as it would if there were a direction $5 \to 4$. A useful description of a graph with N components is given by its *adjacency*

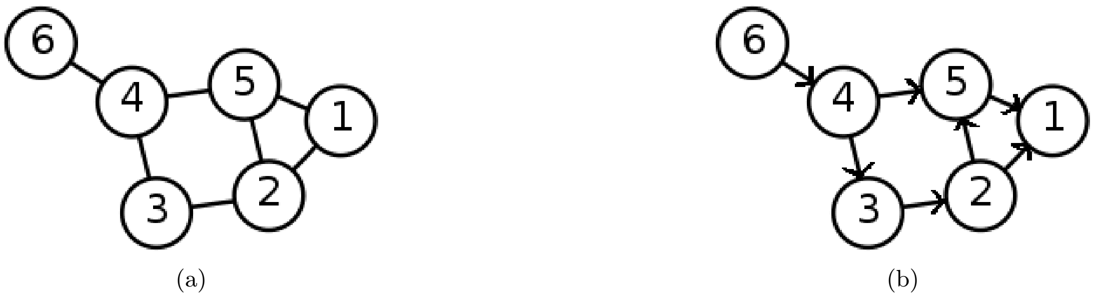


Figure 3.2: (a) A simple 6-element graph and (b) its DAG version

matrix $\mathcal{A}$, whose entry a_{ij} is 1 – or some other weight – if there is an oriented edge between i and j and 0 otherwise. It follows that for undirected graphs, the adjacency matrix is symmetrical, and by convention, if $i = j$ has a nonzero entry, the graph has loops.

Chapter 4

Models

In this Chapter, we present the outline of the underlying models for reasoning about trust and trustworthiness in open systems, which have appeared in the published papers on which this thesis is based. Succinctly, they are practical manifestations of an agoric system, i.e. how information transmission and processing might happen in real situations; the same principle has been utilised in the context of four different frameworks for interaction programmed in two agent development environments (Jason and NetLogo). The first interaction model is based on the concept of bridging low-level and high-level features using a MQTT bridge to Jason and Brahms; The second interaction model improves on this bridging feature, introducing a functional implementation of trustworthiness in open systems, and is again programmed in Jason. The third and fourth interaction models are coded in NetLogo, and deal with fleshing out contradictory information transmission and resolution in a system of sceptic or lazy agents, and with implementing a trust-based distributed ledger allowing for attack strategies by malicious or lazy agents, respectively.

4.1 From Raw Data to Perceptions

The work in the paper [Bottone *et al.* 2016c] considered the problem of connecting real world data exchanged between sensors and actuators with the higher level of abstraction used in frameworks for MAS, identified a gap between the low-level communication mechanisms and the high-level, abstract activities performed in the formalism of multi-agent systems, and developed a practical solution to this problems by means of a software bridge using the MQTT publish-subscribe framework to interface with two BDI agent modelling and programming languages: Jason/AgentSpeak and Brahms. This provides a software link between the low-level communication infrastructure and high-level modelling, simulation and verification environments that can be applied in the context of CPS, IoT and Intelligent Environments.

Basically, in our model, low-level messages in a publish-subscribe infrastructure can give rise to *perceptions* in a multi-agent system and, correspondingly, agents' actions can generate low-level messages and provide feedback. Our solution makes use of an intermediate connector between the publish-subscribe communication layer and the modelling framework for agents coded in Java that implements the MQTT publish-subscribe client and listener, which we describe in detail in Section 5.1. An overview of our bridging solutions is given in Figures 4.1 and 4.2; see Subsections 5.1.1 and 5.1.2 for the corresponding Java code.

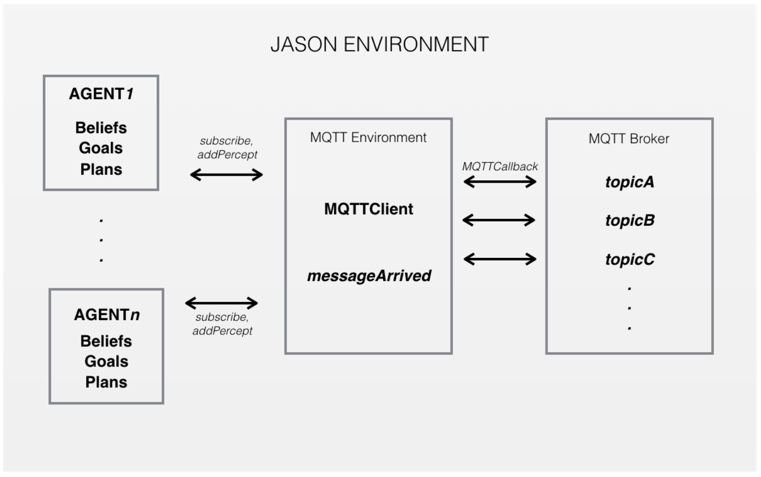


Figure 4.1: Jason bridge

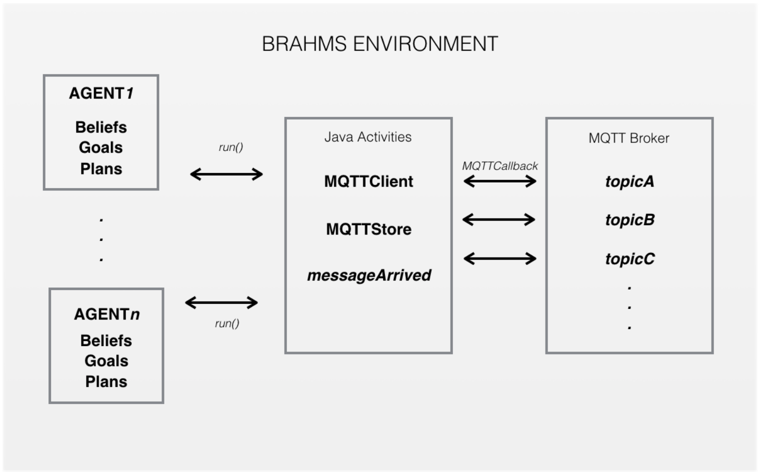


Figure 4.2: Brahms bridge

4.2 Trust in Orchestrated Systems

In the context of trust measurement in CPS, we have mentioned in Section 1.3 that in orchestrated systems, this usually amounts to a dynamic, run-time evaluation and monitoring of variables of interest such as temperature, pressure and depth and extracting features or events. An aspect distinct from simple monitoring of events is the *evaluation* of the *fidelity* of the system of sensors and actuators, especially when these include several types of monitors and users. This links the CPS/IoT paradigm to another crucial aspect of smart environment design, namely their *trustworthiness.* This can be thought of as the assessment of a state or quality – which can be quantitative or qualitative – of engendering trust in the continual functioning of a system. Obviously, it is important not just to be able to collect as much information as possible to optimise functionalities. It is equally essential to overview standards of efficiency and security, by condensing information into one (or a few) simple measures of operational efficiency, to be able to identify malfunctioning and detect possible system intrusions that might become safety critical.

In [Bottone *et al.* 2016b] we presented a theoretical model to characterise the trustworthiness of a fully functional IoT monitoring system by building upon the previous work by [De Florio & Primiero 2015] for software-only systems, and evaluated our approach using a Jason implementation, which is presented in Section 5.2. In particular, we show how to monitor individual sensor values in terms of fidelity functions that trace three distinct behavioural functions *dynamically at run-time* and use the MQTT publish/subscribe model to define controllers that connect the sensors to an actuator. We further define a higher-order function to assess the overall fidelity of the system and induce a trustworthiness evaluation to indicate the running conditions of the system, according to a scale that includes stability and safety.

In [De Florio & Primiero 2015], a notion of fidelity for systems involving software interacting with human users was introduced that relies on the idea that compliant systems manifest a correspondence between domains of actions, yet such a property might not always be perfect. If one is able to monitor and measure the drifting from an ideal total compliance, then one is in a position to assess the level of trustworthiness of said systems. In turn, one can evaluate which safety-security actions are required to maintain the system in working operation.

Let us consider an open system denoted $nOPS$, interacting with n environments, each based on sensors and actuators. $nOPS$ will receive information from the sensors – e.g. temperature and humidity – and send commands to the actuators – for example, the sprinklers. Data from the sensors and activities performed by the actuators are *raw facts* $[r]_i, 1 \leq i \leq n$, accompanied by appropriate binary operations $[\dot{+}]_i$ to form algebraic structures. The *internal representations* of $[r]_i$ by $nOPS$ are expressed by $[q]_i, 1 \leq i \leq n$ internally in its monitoring system, together with operations $[\oplus]_i$ corresponding to $[\dot{+}]_i$. The latter is a somewhat faithful representation of the dynamic variation of the corresponding class of facts. The mapping of raw facts to their representation in $nOPS$ is given by reflective maps Φ_i, bijective functions expressing *perfect fidelity* if and only if the association of $([r]_i, [\dot{+}]_i)$ to $([q]_i, [\oplus]_i)$ is an isomorphism. The shifting from perfect fidelity in the system representation is called *drifting* and it is expressed by the association of an error component Δ_i over time t to what could be called a function ϕ_i of *imperfect fidelity*:

$$\begin{aligned}
&\phi_i : [r]_i \rightarrow [q]_i, \\
&\forall + \in [\dot{+}]_i, \forall \cdot \in [\oplus]_i : \phi_i(r_1 + r_2) = \phi_i(r_1) \cdot \phi_i(r_2) \cdot \Delta_i(t).
\end{aligned}$$

The monitoring of raw facts and their representations can be performed through a client for reading values of associated, asynchronously and continuously updated variables for each of the involved services. In [De Florio & Primiero 2015], such a client is called *Janus*. The role of *Janus* is to retrieve periodically the value of each such variable – such as CPU percentage, memory usage, etc – and store it in the associated memory cells. The values are then compared with "reference behaviours", representing the expected behavioural functions of a trustworthy system. Ideally, such reference values would be those associated with a perfect fidelity function. Such a comparison may be used to detect gradual or sudden behavioural driftings: for example, the former can be associated to progressive deterioration of the system, the latter to instantaneous system breakdown or a takeover.

In [Bottone *et al.* 2016b], we extended the model of *Janus* to a CPS/IoT paradigm. We extended the *Janus* client to retrieve values from sensors and interact with actuators. The state of each component as derived from the activities of the publish/subscribe infrastructure can be compared to a set of behavioural functions, each specifically designed to reflect a fidelity level. The *Janus+Jason* client we wrote in Jason/AgentSpeak is then able to derive an overall evaluation on the system by composing such individual fidelity functions. Trustworthiness is identified as a second order property of the system, induced cumulatively from such functions, and used to assess malfunctioning and required intervention on the IoT platform. Each component c in the platform is assigned a fidelity function: $\phi_c, c \in \{s, a\}$, respectively for a sensor and an actuator, such that $\phi_c : [r]_c \to [q]_c$ expresses the value obtained by mapping the input value from the component's observable behaviour to the preselected expected fidelity function associated to that component. Fidelity is then approximated as the inversely proportional function of the drifting from appropriate mappings ϕ_c. For a platform with two sensors s_i, s_j and two actuators a_i, a_j,

$$\phi_s = 1/f(\Delta(t)_{s_i}, \Delta(t)_{s_j})$$
$$\phi_a = 1/f(\Delta(t)_{a_i}, \Delta(t)_{a_j})$$

for some function f which can be weighted according to domain specific parameters. Each ϕ_c is evaluated as displaying a "high", "medium" or "low" fidelity level when compared to the expected behaviour. The value $\Phi^{nOPS}(t) = \{\phi_s, \phi_a\}$ denotes the global fidelity value for the system parametrised over time. According to the extended theoretical model of *Janus+Jason*, we consider four levels of analysis of fidelity:

1. a *trustworthy system* identifies high levels of $\Phi^{nOPS}(t)$, inducing optimal, sustainable working conditions;

2. an *unstable system* identifies high-to-medium ϕ_s and low ϕ_a levels, meaning that monitoring is well-functioning but interaction with the environment through actuators might be poor, inducing reconfigurable working conditions;

3. an *unsafe system* identifies high-to-medium ϕ_a and low ϕ_s levels, meaning that monitoring is poor, despite the fact that interaction with the environment through actuators might be efficient, inducing unsecure working conditions;

4. an *untrustworthy system* identifies low-levels of $\Phi^{nOPS}(t)$, inducing inadvisable or below-safety working conditions.

Details of the system implementation and code examples are given in Section 5.2.

4.3 The Cost of Trust and Contradictory Information

In this Section, we describe the design of a logical calculus for modelling and resolving contradictory information distributed and transmitted in a network and compute the associated epistemic cost of information when agents identify their communication channels as trustful or distrustful. In the series of papers [Primiero *et al.* 2016] and [Primiero *et al.* 2017] we offered a logical proof model for computing the cost of trust and distrust in a network of agents possessing and sharing contradictory information. In addition to the formal model, a simulation for a variety of network types is also presented whose implementation is described in detail in Section 5.3. It is well known that the processing of trust and distrust in communication channels is also affected, in addition to the agents' raw data and perceptions, by their epistemic attitude and social background. Negative trust, and the intimately related concepts of distrust, mistrust and untrust are important and have found wide application in several academic disciplines, as well as in interdisciplinary works in social networks [Newman 2010, Barabasi 2015] and practical scenarios [van de Bunt *et al.* 2005].

4.3.1 Contradictory Information Transmission

We address two aspects that have been neglected in the literature: (a) in contexts with contradictory information, understanding how positive and negative trust help or hinder the dataflow; and (b) the epistemic costs of (negative) trust transitivity. An efficient model of trust propagation highly depends on the epistemic and structural features of the network in which trust is defined. In particular, understanding the conditions of (dis)trust propagation and the costs related to topological and epistemic factors is crucial both for dynamic network analysis and for access control models, where there is a natural ordering of privileges in a hierarchy. The networks we study are based on basic taxonomies addressed in the network science literature, such as total[1], linear, scale-free and random graph topologies.

In addition to a given network topology, we also assume are that there exists a basic ranking of trustworthiness of agents in the network, such that for example might

[1] Sometimes also called *complete graph* in the literature.

be used in user access control models, and that they have an *epistemic attitude*, i.e. they can be characterised to behave according to well-defined rules in response to new information, such as a requirement to check, verify, or reject new data that comes in.

4.3.2 The Logic (`un`)`SecureND`$^{\texttt{sim}}$

Agents in the logical calculus model underlying (`un`)`SecureND`$^{\texttt{sim}}$ are qualified for simplicity by two representative epistemic attitudes, yet the framework is expressive enough to allow a whole spectrum of attitudes in more complex settings. The two types of agents have the following qualities:

- ▷ *sceptic agents* pay an epistemic cost for performing a checking operation before trusting the received information;
- ▷ *lazy agents* accept without checking information consistent with their current knowledge, while they distrust information inconsistent with current knowledge.

In this context, *positive trust* is a property of the communication between agents required when message passing is executed bottom-up in the access hierarchy, or as a result of a sceptic agent checking information, while *negative trust* is the result of rejecting received contradictory information. Both these situations are associated with *epistemic costs*, which are a way to quantify whether a network that resolves contradictory transmissions by rejecting information is more or less costly than one which facilitates message passing by straightforward acceptance. A memoryless network will discard the resulting determination after every computation each time the network is built; in the published papers, we mostly focused on networks that preserve the memory of previously obtained trusted communications. The logical calculus (`un`)`SecureND`$^{\texttt{sim}}$ is based on the natural deduction calculus `SecureND` devised in [Primiero & Raimondi 2014] as a logic for secure operations on atomic formulas for resources issued by subjects with different privileges guaranteeing that trusted content is checked for consistency at every operation, which was extended by [Primiero 2016] to the specification (`un`)`SecureND` which introduces the semantics for two negative trust protocols, namely the misplacement of trust (mistrust), and betrayal of trust (distrust). The extended logical calculus (`un`)`SecureND`$^{\texttt{sim}}$, which we introduced in [Primiero *et al.* 2016] as a NetLogo simulation environment, makes it possible to simulate contradictory information propagation under trust in a network of ranked agents by studying the logical behaviour of a system using resolution rules and procedures. In this paper we first implemented the algorithms underlying the logical calculus, mapped formal derivations in various graph topologies and published some initial results to experimentally test and evaluate the formal properties. In the follow-up paper [Primiero *et al.* 2017], further refinements and extensions of (`un`)`SecureND`$^{\texttt{sim}}$ in the context of applied network analysis and more comprehensive experimental studies were presented, described in Section 5.3 of this thesis.

As we have mentioned, epistemic attitudes logically encode built-in behaviour of agents and, while superficially rigid in how information transmission should be executed, the logic can be extended to a graded spectrum of epistemic attitudes, as well as to fractional behaviour, i.e. only a percentage of the total number of epistemic agents check information. Agents' operations are those typical of access control models, namely reading (or message reading) and writing (respectively, message passing), enhanced by the following operational semantics:

- ▷ *verification* is required either by a top-down reading operation, i.e., when message passing is executed from below in the hierarchy, or by a reading operation performed by a sceptic agent;
- ▷ *falsification* is formulated as closure of verification under negation and it follows from reading contents that are inconsistent with the current knowledge of the receiver, or from a reading operation performed by a lazy agent;
- ▷ *trust* is a function that follows from verification, when the content passed is consistent with the knowledge of the receiver;
- ▷ *distrust* is formulated as closure of trust under negation and follows from falsification.

Agents are here presented in a contextually empty process of information transmission and their evaluation is purely based on the evaluation of trustfulness and distrustfulness on the basis of criteria of "popularity" – for example, the most trusted or the least distrusted links – and "origin", rather than on criteria such as the inherent, a priori truthfulness of atoms of contradictory information. Thus, our logic, algorithms and simulations focus on the role of trust and distrust as independent from truthfulness criteria, while consistency requirements are crucial.

Adopting a proof-theoretical approach, we define syntactical rules that allow to introduce a given function from premises, and one to eliminate it, i.e., to obtain a conclusion without such a function. This allows us to declare the main properties of the underlying access control model, and some meta-theoretical properties of the related derivability relations and study the validity and role of trust instances. We present the main logical derivations and results as in the extended formulation given in [Primiero *et al.* 2017], while description of the NetLogo implementation and evaluation is deferred to Sections 5.3 and 6.2.

Définition 1 *The syntax of* (un)SecureND$^{\texttt{sim}}$ *is defined by the following alphabet:*

$$V^{<} := \{\mathtt{lazy}(\mathtt{v_i}), \mathtt{sceptic}(\mathtt{v_i})\}$$
$$\phi^V := p^{v_i} \mid \neg\phi^V \mid Read(\phi^V) \mid Verify(\phi^V) \mid Write(\phi^V) \mid Trust(\phi^V)$$
$$\Gamma^V := \{\phi_1^{v_i}, \dots, \phi_n^{v_i}\}.$$

In the above notation, $V^{<}$ indicates the set of lazy and sceptic agents, each denoted by $v_i, v_j, \dots$ The apex means that agents are ordered according to a dominance relation $<$ over $V \times V$; i.e. $v_i < v_j$ means that agent v_i has higher relevance (such as security privileges) than agent v_j. ϕ^V is a meta-variable for boolean atomic formulae closed under negation and functions for reading, writing, verification and trust. Note that the formal model of (un)SecureND$^{\mathtt{sim}}$ and the subsequent algorithms refer to atomic information for simplicity, but the complexity of the formula representing the transmitted information is entirely irrelevant for the results presented here. We also use Γ^{v_i} to express a set of formulae typed by one agent $v_i \in V$, typically the sender, in which a given formula ϕ^V is derivable. Γ^{v_i} is called the *context* in which ϕ^{v_i} is derived[2]. A *judgement* $\Gamma^{v_i} \vdash \phi^{v_j}$ states that a formula ϕ is valid for agent v_j in the context Γ of formulas – including operations – of agent v_j, thus expressing some operation that the agent on the left-hand side of the derivability sign performs on information typed of the agent on the right-hand side of the same sign. When message passing includes more than one agent, this is encoded in the system by an extension of the context, denoted as $\Gamma^{v_i}; \Gamma^{v_j}$. Judgements can also be composed under information transmission procedures. Summarising[3], the following judgements of (un)SecureND$^{\mathtt{sim}}$ are valid for access control given $v_i < v_j$:

- ▷ $\Gamma^{v_j} \vdash Read(\phi^{v_i})$: reading is always allowed when messages come downwards from above in the order relation;
- ▷ If $\Gamma^{v_i} \vdash Read(\phi^{v_j})$, then $\Gamma^{v_i} \vdash Verify(\phi^{v_j})$: messages coming upwards from below in the order relation are passed on under a verification function;
- ▷ If $\mathtt{sceptic}(\mathtt{v_i})$ and $\Gamma^{v_j} \vdash Read(\phi^{v_i})$, then $\Gamma^{v_j} \vdash Verify(\phi^{v_i})$, i.e. message passing is additionally qualified by verification whenever a sceptic agent is on the receiving side and it verifies the information;
- ▷ If $\mathtt{lazy}(\mathtt{v_i})$ and $\Gamma^{v_j} \vdash Read(\phi^{v_i})$, then $\Gamma^{v_j} \vdash \neg Verify(\phi^{v_i})$: when a lazy agent is on the receiving side, information is not verified;

[2]The empty context is denoted by $\cdot \vdash$.

[3]We refer to the original papers [Primiero *et al.* 2016, Primiero *et al.* 2017] for the full syntax and logical implications.

- ▷ If $\Gamma^{v_i} \vdash Read(\phi^{v_j})$ and $\Gamma^{v_i} \vdash \neg\phi$, then $\Gamma^{v_i} \vdash \neg Verify(\phi^{v_j})$: when content read from below contradicts current knowledge, refutation is modelled as negation of verification.

Note that in the last two cases refuted verification leads to negated trust. The complete logical proof system for (`un`)`SecureND`$^{\texttt{sim}}$ is presented in Figure 4.3.

The first rules are for inductive construction of a context Γ^{v_i}. Any such context (or *user profile*) is required to be consistent in that it admits only one of ϕ and $\neg\phi$ after a message passing operation is concluded. The context extension syntax $\Gamma^{v_i}; \Gamma^{v_j}$ indicates that the extension of the profile for agent v_i with a formula ϕ from agent v_j preserves consistency. Inconsistent extensions are not allowed in this proof system. For privilege transfers from above, the rule `read_down` establishes that if a message is owned by a user v_i it can be read downwards. Correspondingly, rule `read_elim` is the elimination rule, i.e., a message that is read (first premise) and preserves consistency (second premise) can be owned, as expressed by the change of label in the formula ϕ. For access from upwards in the dominance relation $v_i < v_j$, the rule `verify_high` says that if a message owned by an agent v_j is read from another agent v_i higher in the relation, then a verification action is required. Similarly, in rule `verify_sceptic` a verification procedure is called when the receiver is sceptic, regardless of whether the receiver is above or below (i.e. independently from the order or reception).

`trust` is the elimination rule for the verification procedure: if `verify` is called and the message preserves consistency (i.e., is derivable in the current agent's profile), then the edge between the agents about that message is trusted. Thus trust elimination corresponds to message passing according to `write_trust`. Note that (`un`)`SecureND`$^{\texttt{sim}}$ models information rejection through a simple consistency checking rule: every message passing operation is eventually accepted if consistent. Rules for verification and trust are not implemented in two cases: in rule `unverified_contra` when messages received conflict with currently held contents; and in rule `unverified_lazy` when a lazy agent is on the receiving side. When two received contents are inconsistent with one another they are resolved via an additional definition based on popularity, i.e. by trust majorisation or distrust minimisation as in the definitions below. Missing verification implies distrust, and thus in turn passing the opposite message by `distrust_elim`. Standard logical statements follow under this proof system. We can then state that the following formal properties of (`un`)`SecureND`$^{\texttt{sim}}$ for the proof system of Figure 4.3 hold:

- ▷ *(Derivation)* Any successful (`un`)`SecureND`$^{\texttt{sim}}$ message passing operation is a derivation tree including a `Write` − `Read` − (`Verify` − `Trust`) − `Write` series of sequents, with the steps `Verify` and `Trust` optional for a lazy agent.

- ▷ *(Satisfiability)* An (`un`)`SecureND`$^{\texttt{sim}}$ judgement $\Gamma^{v_i} \vdash Read(\phi^{v_i})$ is satisfied if there is a derivation D and a branch $D' \subseteq D$ with a final step terminating with such a judgement.

$$\frac{}{p^{v_i} \in \phi^V}\text{ Atom} \qquad \frac{p^{v_i} \in \phi^V}{\neg p^{v_i} \in \phi^V}\text{ }\neg\text{-Intro}$$

$$\frac{\cdot \vdash \phi^{v_i}}{\phi^{v_i} \in \Gamma^{v_i}}\text{ }\Gamma\text{-formation} \qquad \frac{\phi^{v_i} \in \Gamma^{v_i}}{\Gamma^{v_i} \vdash \phi^{v_i}}\text{ premise}$$

$$\frac{\Gamma^{v_i} \vdash \phi^{v_i}}{\Gamma^{v_i}; \Gamma^{v_j} \vdash Read(\phi^{v_i})}\ read\_down$$

$$\frac{\Gamma^{v_i}; \Gamma^{v_j} \vdash Read(\phi^{v_i}) \qquad \Gamma^{v_j} \vdash \phi^{v_i}}{\Gamma^{v_j} \vdash \phi^{v_j}}\ read\_elim$$

$$\frac{\Gamma^{v_j} \vdash \phi^{v_j} \qquad \Gamma^{v_i} \vdash Read(\phi^{v_j})}{\Gamma^{v_i} \vdash Verify(\phi^{v_j})}\ verify\_high$$

$$\frac{\Gamma^{v_j} \vdash Read(\phi^{v_i}) \qquad \texttt{sceptic}(\texttt{v}_\texttt{j}) \in V}{\Gamma^{v_j} \vdash Verify(\phi^{v_i})}\ verify\_sceptic$$

$$\frac{\Gamma^{v_i} \vdash Verify(\phi^{v_j}) \qquad \Gamma^{v_i} \vdash \phi^{v_j}}{\Gamma^{v_i} \vdash Trust(\phi^{v_j})}\ trust$$

$$\frac{\Gamma^{v_i} \vdash Read(\phi^{v_j}) \qquad \Gamma^{v_i} \vdash Trust(\phi^{v_j})}{\Gamma^{v_i} \vdash Write(\phi^{v_j})}\ write\_trust$$

$$\frac{\Gamma^{v_i} \vdash \phi^{v_i} \qquad \Gamma^{v_i} \vdash Read(\neg\phi^{j})}{\Gamma^{v_i} \vdash \neg Verify(\neg\phi^{v_j})}\ unverified\_contra$$

$$\frac{\Gamma^{v_i} \vdash Read(\phi^{v_j}) \qquad \texttt{lazy}(\texttt{v}_\texttt{i}) \in V}{\Gamma^{v_i} \vdash \neg Verify(\phi^{v_j})}\ unverified\_lazy$$

$$\frac{\Gamma^{v_i} \vdash \neg Verify(\phi^{v_j})}{\Gamma^{v_i} \vdash \neg Trust(\phi^{v_j})}\ distrust \qquad \frac{\Gamma^{v_i} \vdash \neg Trust(\phi^{v_j})}{\Gamma^{v_i} \vdash Write(\neg\phi^{v_j})}\ distrust\_elim$$

Figure 4.3: The system (un)SecureND$^{\texttt{sim}}$

▷ *(Validity)* An (un)SecureND$^{\texttt{sim}}$ judgement $\Gamma^V \vdash Read(\phi^V)$ is valid if there is a derivation D and for all branches $D' \subseteq D$ and for all agents $v_i \in V$, there is a final step terminating with such a judgement.

4.3.3 Measuring Trust and Conflict Resolution

Note that each derivation and branch can be analysed in view of its length to count the number of trust rule instances occurring, or equivalently the number of times an atomic message ϕ has been trusted in a given derivation D. This allows us to compute the measure

$$|Trust(\phi^V)|_D = |Verify(\phi^V)|_D$$

for all $v_i \in V$. Conversely, one can compute the number of times an atomic message ϕ has been distrusted in a given derivation D:

$$|Distrust(\phi^V)|_D = |\neg Verify(\phi^V)|_D$$

for all $v_i \in V$. These two computable measures give rise to simple conflict resolution formulas by trust majorisation or distrust minimisation for the case in which consistency fails, namely

- ▷ (Conflict Resolution by Trust Majority) Given a derivation D_1 terminating in $\Gamma^{v_i} \vdash Write(\phi^{v_i})$ and a derivation D_2 terminating in $\Gamma^{v_j} \vdash Write(\neg\phi^{v_j})$, a new step holds which takes as premises $\Gamma^k \vdash Read(\phi^{v_i})$ and $\Gamma^k \vdash Read(\neg\phi^{v_j})$ respectively, and concludes $\Gamma^{v_k} \vdash \phi^{v_k}$ if and only if

$$|Trust(\phi^V)|_{D_1} > |Trust(\neg\phi^V)|_{D_2}$$

- ▷ (Conflict Resolution by Distrust Minority) Given a derivation D_1 terminating in $\Gamma^{v_i} \vdash Write(\phi^{v_i})$ and a derivation D_2 terminating in $\Gamma^{v_j} \vdash Write(\neg\phi^{v_j})$, a new step holds which takes as premises $\Gamma^k \vdash Read(\phi^{v_i})$ and $\Gamma^k \vdash Read(\neg\phi^{v_j})$ respectively, and concludes $\Gamma^{v_k} \vdash \phi^{v_k}$ if and only if

$$|DisTrust(\phi^V)|_{D_1} < |DisTrust(\neg\phi^V)|_{D_2}$$

Such conflict resolution rules say that one can resolve the conflicting merges at any stage where branch merging occurs, either by (a) preserving the most trusted content, or (b) preserving the least distrusted content. By the validity property of (un)SecureND$^{\texttt{sim}}$ above, one logically concludes that for each derivation D with a valid formula $\Gamma^V \vdash \phi^V$ there exists a graph $\mathcal{G}$ that is unanimously labelled by the atomic formula ϕ. Full proofs and derivations of structural properties corresponding to the application of all the rules in (un)SecureND$^{\texttt{sim}}$ listed in Figure 4.3 are provided in the original paper [Primiero *et al.* 2017].

4.4 Block-free Distributed Ledgers

In this Section, we focus on the nature of distributed ledgers, and in particular the mathematical objects underlying the so-called Tangle [Popov 2017], as *accumulated information flow*. In Subsections 4.4.2, 4.4.3 and 4.4.4 we describe its mathematical structure, attachment and consensus model and update rules and strategies; we defer to Section 5.4 the description and implementation of an extensible, open-source multi-agent simulation environment for such structures built in NetLogo and provide results in context in Subsection 6.3.1.

Credit money [Simmel 1907], is a social mechanism which has evolved to transfer trust between economic agents and whose usefulness arises as a network externality that facilitates transactions [Kocherlakota 1998, Nash Jr 2002]. The traditional roles of money – or currency – are that of a unit of account, a store of value, and a medium of exchange, and how money arises is still the subject of academic research [Graeber 2011, Yasutomi 1995]. Cryptocurrencies attempt to replicate this memory function and other characteristics of real-world money such as safety and consistency. Early attempts at cryptographic cash relied on trusted authorities that maintained centralised ledgers, such as banks and credit card companies. The main technological advance of Bitcoin [Nakamoto 2008] was to introduce a distributed ledger secured by the majority rule without any central authority, by means of a so-called *blockchain* and standard cryptographic primitives like signatures and hash functions, and his seminal paper spurred academic and practical interest. It also introduced the possibility of transaction scripting as a way of enabling smart contracts and micro-transactions based on distributed architectures which are suitable for CPS and the IoT.

The blockchain used by Bitcoin and its descendants is a time-linear data structure: transaction data is stored in *blocks* which must each contain a reference to the block that came before it. Blocks are created by specialised users called *miners* that perform a cryptographic proof-of-work (PoW). It is natural to consider the temporal succession of blocks as a directed flow of transactions linked together by consensus. The blockchain algorithm has proved resilient to attacks and double spending, suffering only setbacks in coin exchanges. However, as a cryptocurrency for lightweight systems in the IoT, Bitcoin has two serious drawbacks which make it unsuitable for micro-payments; firstly, because of the dependence on miners for processing and verification, the transaction fees are relatively high and rising, especially in low throughput; secondly, it scales poorly since the blockchain network performance degrades in the number of users. Thus, Bitcoin suffers from scaling and fee issues are intimately connected to the mining system and its incentive structure, which under consolidation of miners, reduce the degree of decentralisation of the network. *Mining pools* now make up over 90 percent of the hashpower in the Bitcoin network, and tend to be heavily concentrated geographically. Verification delays for reaching consensus are also commonplace: because of the fixed

MB limit on the blocks, Bitcoin currently processes 3/4 transaction per second (txps) and is capped at 7 txps, while Visa and MasterCard are capable of processing 60000 txps. Security of the protocol has also been called into question when miners can collude or are exposed to geopolitical risk.

4.4.1 DAG-based Cryptocurrencies

A promising avenue of recent research and development has involved blockchain-free currencies. In these approaches, the blockchain and its consensus algorithm are replaced by a directed graph of cross-verifying transactions based on the mathematical properties of a *Dyrected Acyclic Graph (DAG)*, which serves as a truly distributed ledger and, generally speaking, reaches consensus by accumulation of information about the state of the network. The essential idea is that to issue a transaction, users of the distributed ledger must work to approve other transactions, thus checking for conflicts and double spending, and when a transaction receives additional approvals by the chain of ensuing transactions, it becomes accepted by the system with a high degree of confidence. In a DAG, each node represents a transaction and each edge a reference, or approval, of some other transaction in a specific direction. Such graphs are usually built up from an initial parent root called the *genesis* transaction and evolve according to precise rules, representing in this sense a lightweight generalisation of blockchain (for an alternative construction where the direction of approval is reversed from parent to child transactions, see [Boyen *et al.* 2017]). Several DAG-based cryptocurrencies have been recently independently proposed and implemented, among them RaiBlocks [LeMahieu 2017], DagCoin, Byteball [Churyumov 2015] and Iota [Foundation 2017], which deviate from each other in the details of implementation and consensus protocols. RaiBlocks achieves consensus by using a deterministic block-lattice structure where each account has its own balance-weighted blockchain which resembles the account's transaction and balance history, and can only be updated by its owner, similar to SPECTRE [Sompolinsky *et al.* 2017] with restrictive permissions. Byteball reaches consensus by using a main chain of honest, trusted witnesses that reference one or more previous transactions via a Markov Chain Monte Carlo (MCMC)[4] selection procedure for referrals. Iota's consensus model is based on the cumulative PoW of stacked transaction where two previous transactions with low weight are selected. The latter implementation, based on the mathematical construct called the *Tangle* [Popov 2017], is particularly simple to describe, yet flexible and robust to use, and has several attractive features that make it well suited for CPS and IoT. More generally, as the results of [Sompolinsky *et al.* 2017] on the DAG block voting mechanism imply, the way consensus is achieved, despite the combinatorial explosion in transaction messages, can be

[4]A MCMC type of algorithm [Robert & Casella 2005] samples from a probability distribution, by constructing a Markov chain that has the desired distribution as its equilibrium distribution, i.e. one which is expected to converge to a desired distribution after a number of steps.

engineered in such a way that consensus is pruned and maintained by various forms of optimisation to scale to todays' and future centralised payment processing throughput, and an agoric perspective provides a natural solution to this challenge by virtue of its focus on efficient information processing.

4.4.2 The Tangle Process

The Tangle introduced by [Popov 2017] can be thought of as a dynamic process on the space of oriented, rooted DAGs, which grows in time according to a Poisson clock for the flow of arrivals where new nodes (i.e., new transactions) are continually attached to the graph at locations which are chosen according to specific rules, and no nodes or edges are ever deleted. More specifically, the Tangle is a graph $\mathcal{T} = (V, E)$, where as customary V is the set of nodes and E the set of edges and for each $v \in V$, the in-degrees and out-degrees are specified by $d_{\mathbf{in}}(v) = \sharp\{e = (c_1, c_2) \in E : c_2 = v\}$ and $d_{\mathbf{out}}(v) = \sharp\{e = (c_1, c_2) \in E : c_1 = v\}$, with $\sharp$ denoting set cardinality. For $v_1, v_2 \in V$, v_1 *approves* v_2 if $(v_1, v_2) \in E$, written $v_1 \nrightarrow v_2$. Let $\mathcal{A}(u) = \{v : (u, v) \in E\}$ be the set of nodes approved by u. If there exists a covering chain $u = v_0, v_1, \ldots, v_k = v$ such that $v_j \in \mathcal{A}(v_{j-1})\, \forall j = 1, \ldots, k$ forms a directed path between them, we say that u *indirectly approves* v. We call the set of nodes such that $\{v : d_{\mathbf{in}}(v) = 0\}$ the set of *tips*. The following additional rules apply:

(a) within the set of all possible DAGs, each graph $T \in \mathcal{T}$ is finite and with out-degree edge multiplicity at most 2, i.e. $\forall v \in V, d_{\mathbf{out}}(v) \leq 2$;

(b) there exists a *genesis* root $\rho \in V$ such that $d_{\mathbf{out}}(\rho) = 0$ and $d_{\mathbf{out}}(v) = 2\, \forall v \in V \backslash \{\rho\}$;

(c) any other node $v \in V$ references ρ, i.e. there is an oriented path of approvals from this node to the genesis ρ; and

(d) there are *no cycles*, i.e. paths of the type $v = v_1, \ldots, v_k = v$ for any v and k.

From the assumptions above, it necessarily follows at any time t the state of the Tangle $\mathcal{T}(t)$ can be concisely described by a sparse *adjacency matrix*, which is strictly lower triangular; thus the state of this matrix over time is given by a first row of 0s, a number of rows with 1s in the first column for each genesis transaction, and rows of two 1s to the left of the diagonal thereafter. This can be efficiently stored in *adjacency lists*, namely a collection of nodes and nonzero edge positions [Wikipedia 2018a]. More generally, the Tangle is a continuous-time stochastic process on the space $\mathcal{T}_\infty = \cup_1^n \mathcal{T}_i \cup \mathcal{T}_{n+1} \cup \ldots$ with initial state given by $V_{\mathcal{T}}(0) = \rho$, $E_{\mathcal{T}}(0) = \emptyset$ and evolving according to the following rules:

- As a result of the flow of new transactions, the Tangle grows in time, i.e. for any two times $0 \leq t_1 < t_2$, $V_{\mathcal{T}}(t_1) \subset V_{\mathcal{T}}(t_2)$ and $E_{\mathcal{T}}(t_1) \subset E_{\mathcal{T}}(t_2)$.
- For a fixed mean transactions per unit time $\lambda > 0$ the Poisson process $\Lambda(t) := \mathrm{Pois}(\lambda)$ gives the incoming transactions that then attach to $\mathcal{T}(t)$.
- Each transaction chooses two nodes v_1, v_2 and attaches a new node v to $\mathcal{T}$ with oriented edges $v_1 \to v$ and $v_2 \to v$, so that each tip unites to the set of nodes and edge points of $\mathcal{T}$ (two-edge-multiplicity rule).

This kind of DAG-based process can also be generalised to unary or n-ary out-degree multiplicities, and we can equip nodes with quantitative attributes such as attitudes or other characteristics.

4.4.3 Consensus by Cumulative Weight

The Tangle achieves consensus by attaching to every transaction a positive integer within some specified bounds and time limits[5], which describes a trustworthiness property of a transaction. The basic idea is that a transaction with a higher weight is more important than a transaction with a lower weight in deciding attachment. With this in mind, we can define on $\mathcal{T}$ the partial order with respect to approvals:

$$\mathcal{P}^x_{(t)} = \{y \in \mathcal{T}(t) : y \looparrowleft x\}$$
$$\mathcal{F}^x_{(t)} = \{z \in \mathcal{T}(t) : z \looparrowright x\}$$

such that we successively refine (or zoom into) a past $\mathcal{P}_{(t)}$ and expand to a future $\mathcal{F}_{(t)}$ with respect to node x at a time t when they become attached. The consistency and tip selection procedure works as follows. Define the *cumulative weight* of node x as $\mathcal{H}^x_{(t)} = 1 + \sharp\{\mathcal{F}^x_{(t)}\}$, which increases with the number of nodes that directly or indirectly reference it. For any $t > 0$ if $y \looparrowright x$ then one necessarily has $\mathcal{H}^x_{(t)} - \mathcal{H}^y_{(t)} \geq 1$, which implies that $\mathcal{H}^y_{(t)} = 1$ if y is a tip. We say that a transaction gets *confirmed* when it reaches a threshold θ which is sufficiently high when relative to the network usage and load. In real-world networks such as those used in the Iota platform, each incoming node gets to decide which transaction gets orphaned or approved, thus propagating the Tangle in time; see `tangle.glumb.de` for a visualisation showing the consensus model based on live transaction data. It is evident that one can without loss of generality assume the Markov property, as each successive state of the process will depend only

[5]Currently, Iota uses 3^n, but for simplicity one can assume that each node has weight 1 to start with.

on the current state of the Tangle. In the Iota whitepaper [Popov 2017], it is suggested that the optimal growth is obtained by evaluating some statistics about the transactions, which basically amount to updating their cumulative weights. Ideally, we would like the graph to grow so that, eventually, all issued transactions are confirmed, according to some "optimal" criterion.

4.4.4 Tip Selection Strategies

Let $\mathcal{L}(t)$ be the set of all vertices that are tips in the tangle at time t, and let $L(t) = \sharp\{x \in \mathcal{T}(t) : \mathcal{H}^x_{(t)} = 1\}$ be its cardinality. Note that in general, $\mathcal{L}(t)$ can be decomposed into two sets of *visible* and *hidden* tips due to network delays, that is from the viewpoint of an agent that is trying to approve previous transactions, only a subset of the total number of tips extant at a given moment may be visible in a list of available tips.

Ideally, one would like the stochastic processes for both the Tangle and $\mathcal{L}(t)$ to be well behaved in the limit of a large number of transactions. In [Popov 2017], theoretical considerations are advanced for $\mathcal{L}(t)$ to be positive recurrent as $t \to \infty$, i.e. $\mathbb{P}[L(t) = k] > 0$ for $k \geq 1$, rather than transient or escaping to infinity, which would leave many unapproved transaction orphaned. In practice, little is assumed in the implementation of the distributed ledger, apart from the strict approval rule, i.e any new transaction must reference two other transactions (tips) already in the Tangle. It is entirely possible that two transactions, each posted by different users, or by the same user in a short time frame, reference the same tips as each other; and in fact this happens all the time because of network latency[6]. A node can choose tips in any way it finds convenient. A particularly lazy node might try to approve a fixed pair of very old transactions, without penalty, thus not contributing to the tip approval process and increasing the likelihood that some of these might be orphaned. "Anything goes" effectively renders the Tangle a random graph.

Random Tip Selection. The simplest strategy is for each new node to select two tips uniformly at random from the list of available tips, and approve them. While conceptually simple, this strategy has the disadvantage that it does not sufficiently protect against lazy or malicious nodes. Under this hypothesis, in the steady state $L(t)$ should fluctuate around $L_0 = 2\lambda\Delta t$ in any interval Δt. The drawback of this strategy is that, especially for low network load, it can lead to *opportunistic* behaviour in that lazy nodes may decide to repeatedly select the same transaction to attach new ones, thus reducing the overall number of tips. This kind of structure can often be seen in Tangle visualisers. It can also lead to *malicious* behaviour, where users try to use their own algorithm to select tips to spam the network – artificially inflating the number of tips by issuing many transactions that approve a fixed pair of transactions – and take

[6]One tip can be selected in good faith by two different nodes at the same time until "snapshotting", i.e. persistent storage.

it over, or attempt a double spend of funds – growing for example a parasite chain with the usual attachment rules, and then attaching to the Tangle a "conflicting" transaction.

MCMC Selection Algorithm. A more sophisticated strategy to encourage "optimal" growth of the Tangle is to use a particle filter or Markov Chain Monte Carlo (MCMC) algorithm to select two tips on the Tangle [Popov 2017, Popov *et al.* 2017]. This still selects at random, but introduces a bias towards "honest tips" by means of an exponentially tilted random walk on the nodes of $\mathcal{T}(t)$, which become the sites of walkers that walk the reverse-directed links from the genesis ρ (or any other cutset with equal cumulative weight) to the tip. Note that while the out-degree of a node is fixed, the in-degree has a distribution, and each node may use its own pseudo-random number generator to simulate the walks. Essentially, the typical MCMC strategy selection algorithm will be alongside the lines of:

1. Choose a cutset, or suitable interval of nodes in chronological order. Usually the particle walk starts at ρ or somewhere else deep in the Tangle; if the walk does not start at the genesis, we set $q \geq 0$ as the backtracking parameter.

2. Independently place N particles on that cutset.

3. Let them perform a random walk, with a transition from x to y only possible if y approves x. (Optionally, repeat the walk, or select a high number of walkers N, if the two selected tips are not distinct).

4. The transition probabilities $\mathbb{P}^f_{xy}$ between two nodes sharing a directed edge $x \hookleftarrow y$ are proportional to some monotone, non-increasing function f of the difference in cumulative weights $\mathcal{H}^x_{(t-h)} - \mathcal{H}^y_{(t-h)}$[7]. The particle is stopped when it hits a node $v \in \mathcal{L}(t-h)$[8]. Usually, $f(s) = \exp(-\alpha s)$ with $\alpha \geq 0$ having the meaning of inverse temperature (or measure of randomness). For any y and x one has the Boltzmann-Gibbs distributions:

$$\frac{(1-q)\exp\left[-\alpha\left(\mathcal{H}^x_{(t-h)} - \mathcal{H}^y_{(t-h)}\right)\right]}{\sum_{z:x\in\mathcal{A}(z)}\exp\left[-\alpha\left(\mathcal{H}^x_{(t-h)} - \mathcal{H}^z_{(t-h)}\right)\right]} \qquad x \in \mathcal{A}(y) \tag{4.1}$$

   where $\mathcal{A}(\cdot)$ is as defined previously.

Intuitively, such a strategy spreads the approval process evenly along the most recent tips in the Tangle. If α approaches 0, this strategy is equivalent to the uniform random selection strategy; for high α it will tend to pseudo-deterministically assign high probabilities to fixed paths indexed by the highest cumulative weight, and the

[7]In general, the node that issues the transaction might only know the state of the tangle network with a delay, $\mathcal{T}(t-h)$.

[8]If $h > 0$, it might not even be a tip anymore.

number of tips will grow linearly in each time step. A transaction is confirmed with a sufficiently high confidence level $\iota_0 \approx 1$ if in the low temperature regime, the walk ends in a tip that references the transaction. The rationale for choosing this type of selection strategy compared to a simpler random tip choice is that in a well-behaved Tangle, the hashing power of the network – measured by large increases in cumulative weight – is higher than that of an attacker that tries to attach a long parasite chain of transactions, whose cumulative weights would necessarily be much smaller than the sites they reference, and thus parasite sites would have a low transition probability from the main sites of the Tangle. See Section 4.1 of [Popov 2017] and [Popov *et al.* 2017] for a game-theoretic justification.

Note that the Tangle $\mathcal{T}(t)$ as constructed induces a continuous time transient Markov Chain with large state space even for a fixed time t. This means that the corresponding adjacency and transition matrices suffer from combinatorial explosion, and expanding access times[9].

4.4.5 Mean Tip Approval Times

Also by assumption, the Poisson clock leads to exponentially-distributed inter-arrival times between consecutive transactions: if we hypothesise λ to be the number of new arriving transactions per unit time, the mean arrival time will be on average $\frac{1}{\lambda}$ and will have a waiting-time of 0 as its mode. Thus[10] the "wider" the Tangle $\mathcal{T}(t)$ is – i.e. the larger it scales in terms of the incoming transactions and number of users – the more instantaneous we can expect the Tangle propagation to be. Straightforward statistical analyses of the public data from [Foundation 2017] confirm this to be the case.

[9]The sparse adjacency matrix for $10,000$ nodes, for example, requires 500MB, and 3GB for 25000, which is still manageable on a 2015 vintage laptop but unwieldy for a large number of sequential writes.

[10]Disregarding for simplicity the proof-of-work nonces and network latency issues, which can be nevertheless be modelled by a compound clock as the average number of revealed tips will be $\lambda \cdot h$ for h the network delay in seconds, which is a constant.

Chapter 5

Implementation

In this Chapter, we describe the implementation details of the four modelling frameworks described in the previous Chapters. In particular, Section 5.1 presents the MQTT-Jason bridges written for the published papers [Bottone *et al.* 2016c], [Bottone *et al.* 2016b] and [Bottone *et al.* 2016a]; Section 5.2 describes how the orchestration of information flow in a CPS might be leveraged to attain a trustworthiness function of the whole system to enable real-time monitoring; Section 5.3 presents a simulation of information transmission and epistemic costs of contradictory information in a network; and Section 5.4 introduces a simulation environment for multi-agent modelling and sampling of DAG-based cryptocurrency transactions.

5.1 Implementing Bridges

In this Section we present three implementations of software bridges between multi-agent simulation environments and the publish-subscribe communications protocol MQTT, corresponding to the diagrams in Figures 4.1 and 4.2.

5.1.1 Connecting MQTT with Jason

Jason is an example of multi-agent framework in which modellers have access to the execution engine in such a way that the environment (and thus, the beliefs of the agents) can be modified by means of Java code. Specifically, a new **Environment** can be created by subclassing the Jason class `Environment`. Figure 5.1 reports excerpts from the implementation of a bridge between a MQTT infrastructure and Jason. The new class `MQTTEnvironment` subclasses the default Environment class and extends the initialisation method (line 3) with the connection to a MQTT broker and the subscription to appropriate topics (lines 9 and 10). The key method here is `messageArrived` on line 16: this method is invoked when a message arrives from the broker. The message is parsed appropriately with the method `parse_message` and a new *percept* is created in Jason (line 18). This percept may result in a new belief in an agent and give rise to new intentions. This environment has been employed to monitor the trustworthiness of a software infrastructure for an air conditioning system, as described in the ensuing Subsubsection 5.2.3.1, scaling to several hundreds of sensors. We also provide further commentary of the performance of this system in Section 6.1.

Similarly, the **Environment** can be extended in a very simple way with new actions that can be invoked by agents when they want to publish a MQTT message. The code for this example is available online at `https://bitbucket.org/bottone/mqttbridges`; we refer to it for additional details.

```
public class MQTTEnvironment extends Environment implements MqttCallback {

    public void init(String[] args) {
        // [...]
        try {
            client = new MqttClient(MQTT_BROKER, "JasonMQTTEnvironment");
            client.connect();
            // [...]
            client.setCallback(this);
            client.subscribe("TOPIC/#");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void messageArrived(String topic, MqttMessage message) throws Exception {
      // [...]
      addPercept(Literal.parseLiteral(parse_message(topic,message)));
      // [..]
      }
    }
```

Figure 5.1: Jason - MQTT bridging environment.

5.1.2 Connecting MQTT with Brahms

In contrast to Jason, Brahms is an archetype of a framework in which developers are not allowed to modify the *state* of agents directly. This means that it is not possible to create new beliefs automatically whenever a new MQTT message is received. To address this issue we have created an intermediate *store* for MQTT messages, which should be run in parallel with Brahms. Even if beliefs cannot be created automatically, as described above Brahms can be extended with Java actions that can be invoked by agents. The key idea is to use these actions to query the external store. The latter, in turn, acts as a buffer between the MQTT broker and the asynchronous actions of Brahms' agents.

Figure 5.2 reports excerpts of our implementation (the full code is again available at `https://bitbucket.org/bottone/mqttbridges`). Notice that, differently from the case of Jason described above, this code now runs independently from Brahms and it is not an extension of any of the Brahms classes. Similarly to the Jason Environment, the class `MQTTStore` implements the interface `MqttCallback` to be able to subscribe to messages (line 1). This new store starts a TCP socket (defined in line 3, constructor code omitted but available online). The constructor method starting at line 9 establishes a connection with the broker and subscribes to appropriate topics. It also creates an empty list of MQTT messages. Messages received from the broker and corresponding to the appropriate topic are added to this list using the method `messageArrived` (lines 25 to 28). The method `run` is the method invoked by the TCP server whenever a new connection is made to this class. Essentially, this is the standard method invoked by

```
 1 public class MQTTStore implements MqttCallback {
 2
 3   ServerSocket incomingSocket;
 4   private MqttClient mQTTClient;
 5   // Incoming MQTT messages are stored here
 6   LinkedBlockingQueue<String> mQTTMessages = null;
 7   // [...]
 8
 9   public MQTTStore() {
10     // [...]
11     mQTTClient = new MqttClient(BROKER_ADDRESS, CLIENT_NAME);
12     mQTTClient.connect();
13     mQTTClient.setCallback(this);
14     mQTTClient.subscribe(SUBSCRIBED_TOPIC+"/#");
15     // [..]
16     mQTTMessages = new LinkedBlockingQueue<>();
17     // [...]
18   }
19
20   public void run() {
21     // Here: wait for connections and return the
22     // list of MQTT messages received
23   }
24
25   @Override
26   public void messageArrived(String topic, MqttMessage message) {
27     mQTTMessages.add(topic+" "+new String(message.getPayload()));
28   }
29 }
```

Figure 5.2: Brahms - MQTT bridging environment.

Brahms Java activities to send and receive messages. To send a message, a Brahms activity needs to provide a topic and an actual message. When a Brahms activity requests the stored messages, the content of the list `mQTTMessages` (line 6) is returned and the list is emptied. This approach makes the class `MQTTStore` a buffer between the broker and the Brahms running instance, given that it is not possible to implement `MqttCallback` directly in Brahms.

5.1.3 Implementing Virtual Pheromones with MQTT and Jason

Stigmergy is a biological mechanism of spontaneous, indirect coordination between agents, where the trace left in the environment by an action stimulates the performance of a subsequent action, by the same or a different agent. It was originally proposed for social termites [Grassé 1959] and has found wider application outside social biology [Holland & Melhuish 1999, Bonabeau 1999, Beni 2004].

Note that stigmergic markers need not correspond to physical traces; the concept translates easily to virtual particles in the environment. In the work presented in the paper [Bottone *et al.* 2016a], we implemented a virtual pheromone system based on

```
 1 public class SimpleMQTTSubscriberAndPublisher implements MqttCallback {
 2
 3   private MqttClient client;
 4   private static String BROKER_ADDR="tcp://broker.hivemq.com:1883";
 5   private static String CLIENT_NAME="TestStigmergyBDI";
 6
 7   // The topic we are interested in.
 8   private static String SUBSCRIBED_TOPIC="MQTTTest";
 9
10   public SimpleMQTTSubscriberAndPublisher() {
11
12     // Connect to the broker:
13     client = new MqttClient(BROKER_ADDR, CLIENT_NAME);
14     client.connect();
15
16     // Set callback for subscriptions
17     client.setCallback(this);
18     client.subscribe(SUBSCRIBED_TOPIC+"/#");
19
20     // Publish a message under a topic
21     MqttMessage message = new MqttMessage();
22     message.setPayload("Hello World".getBytes());
23     client.publish("pubTopic/subtopic", message);
24   }
25 }
```

Figure 5.3: Java MQTT client (excerpts).

the bridging integration of MQTT and Jason that can be used for modelling swarm robotics scenarios and to deploy complex and robust coordination across a variety of robotic platforms with limited computational resource overhead, exploiting the rapid increase in data transfer rates to offload tasks without hard real time requirements.

The Java code listed in Figure 5.3 shows how to create a client that connects to a public broker (line 14), subscribes to a topic (line 18), and sends a message (lines 21 to 23). When a message whose topic matches the one specified on line 8 is generated, a listener (line 17) is invoked. In [Barbon *et al.* 2016], the potential of implementing MQTT on resource-constrained devices such as Arduino and Raspberry Pi was explored. In this work, we exploited the features offered by MQTT to dispatch and distribute information about the virtual pheromone map shared by the robots embedding a Raspberry Pi platform.

As before, a new *Environment* can be created by subclassing the Jason class `Environment`, and the implementation details are very similar with minor changes to the callback method, so that we refer again to Figure 5.1 from Subsection 5.1.1 for excerpts of the bridging between a MQTT infrastructure and Jason. The new class `MQTTEnvironment` subclasses the default Environment class and extends the initialisation method (line 3) with the connection to a MQTT broker and the subscription to appropriate topics (lines 9 and 10). The key method here is `messageArrived` on line 16

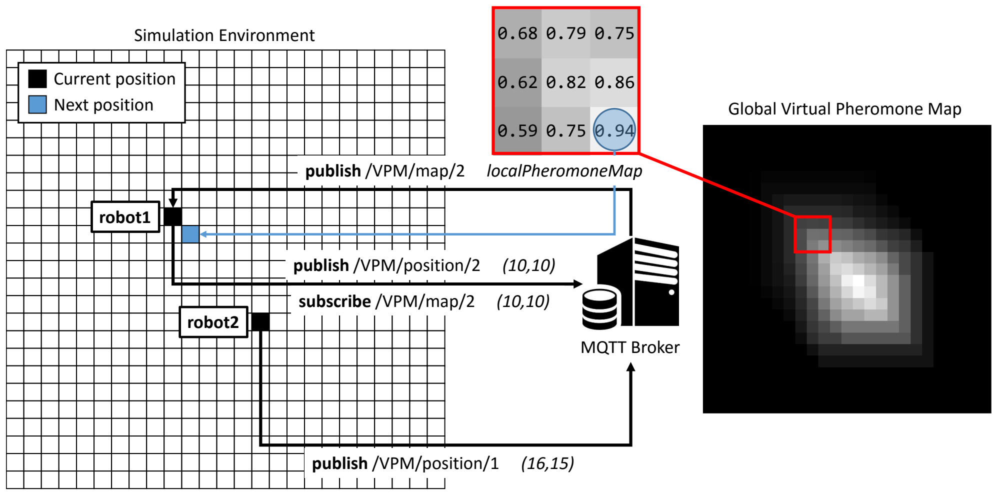


Figure 5.4: An example of how two BDI robots interact with the MQTT Broker in order to send their positions to the back-end server and retrieve the local pheromone map (the `mark` structure) around their position.

that is invoked when a message arrives from the broker. The message is appropriately parsed and a new *percept* is created (line 18). This percept may result in a new belief in an agent and gives rise to new intentions. Similarly, the Environment can be extended in a very simple way with new actions that can be invoked by agents when they want to publish a MQTT message.

5.1.3.1 Implementing Stigmergy in Jason

A stigmergic environment can be created by extending the default environment `GridWorldModel`, following the approach presented in [Barbieri & Mascardi 2011]. In particular, in the Java implementation of the *Environment*, we add a private field holding the intensity of pheromones in each cell of the grid. The intensity of pheromones in the grid can be updated in two ways:

- ▷ (a) Directly by one of the agents: implementing a new internal action `mark` in the Java environment to send the appropriate MQTT message to all the agents.
- ▷ (b) Via MQTT: the message is delivered to the Java environment by the callback method described in Figure 5.3.

Agents implemented in Jason can get a local map of the intensity of pheromones in their neighbourhood by invoking a new internal action `getMarks` implemented in the environment. The local map is an array of nine triples in the form `(x,y,i)`, where `i` is the pheromone intensity, centred around the current position of the agent.

The main difference between our approach and the one presented in [Barbieri & Mascardi 2011] is that in our case we have *multiple instances* of Jason agents, one per robot, running in parallel. Moreover, their approach relies on a local server for messaging, while we employ MQTT as our communication mechanism. This has the additional benefit of allowing the incorporation of agents implemented in other frameworks, provided that they support MQTT messaging. Figure 5.4 shows an example of how two BDI robots interact with the MQTT Broker in order to send their positions to the back-end server and retrieve the local pheromone map (the `mark` structure) around their position. All the robots present in the environment, equipped with their own localisation system and wireless communication, subscribe to the relative topics, where the local pheromone map will be received: `/VPM/map/robotID`. At each time step, robots send their position to the back-end server publishing their coordinates to the MQTT topic: `/VPM/position/robotID`. Then, the back-end server updates the global virtual pheromone map and publishes the local pheromone map to each robot by means of the dedicated topics `/VPM/map/robotID`. Finally, the robots choose the new position among the nine possibilities as the pheromone coordinates of the point with the higher intensity (`i`). In the absence of a local pheromone map, robots choose the new position randomly.

The update process of the global pheromone map is based on the potential field model [Susnea 2015]. At each time step, it computes the intensity at distance d_k from each pheromone k using equation 5.1:

$$p(d_k) = \begin{cases} p_k\left(1 - \dfrac{d_k}{\sigma}\right) & \text{if } 0 < d_k < \sigma \\ 0 & \text{if } d_k \geq \sigma \end{cases} \tag{5.1}$$

where $p(d_k)$ is the intensity of pheromone k at distance d_k due to diffusion, σ is the sensitivity range, and p_k is the actual intensity of pheromone k. Due to the stigmergic aggregation of all the N sources located within σ, the resulting pheromone intensity sensed in an arbitrary location is given by equation 5.2:

$$P = \sum_{k=1}^{N} p_k \left(1 - \frac{d_k}{\sigma}\right). \tag{5.2}$$

Assuming that the evaporation effect decreases the pheromone intensity linearly, it is possible to update the resulting pheromone at time t as shown in equation 5.3:

$$P = \sum_{k=1}^{N} p_k \left(1 - \frac{d_k}{\sigma}\right)\left(1 - \frac{t - t_k}{\tau}\right) \tag{5.3}$$

where t_k is the time of creation of the pheromone k and τ is the evaporation parameter. The value of P around the current position of the robot is returned to it as a message (*localPheromoneMap* in Figure 5.4) published on the previously subscribed topic `/VPM/map/robotID`, and thereafter the robot acts as if it has its own pheromone sensing system [Susnea *et al.* 2009], choosing the next position as the maximum of the local pheromone map (the blue square in Figure 5.4).

5.2 Monitoring Trust

In this section we describe how we encoded the fidelity functions in Jason for the paper [Bottone *et al.* 2016b]. The overall setup is based on a MAS where each component is a separate agent, introducing a special agent – the controller or *Janus+Jason* – that oversees and evaluates the overall fidelity of the system.

5.2.1 The Orchestrating System

Within the *Janus+Jason* controller, we further define higher-order plans to assess the overall fidelity of the system and induce a trustworthiness evaluation to indicate the system's running conditions, according to the fidelity scale including stability and safety conditions introduced in Section 4.2. The software has been open sourced, with the Jason and Java files available at `https://bitbucket.org/mdxmase/janus-jason/`.

5.2.2 Environment and Communications Protocols

The default Jason environment comes with built-in classes that make use of the `.send` and `.broadcast` internal actions for single and multiple inter-agent communication, respectively. In addition to this infrastructure, we built a custom environment `MQTTEnvironment` to bridge between MQTT messages and Jason. The Java class `MQTTEnvironment` does two things: it implements the `MQTTCallback` interface for MQTT connection/disconnection and it sets up the topics in the publish/subscribe format so that when a message is published, it adds the relevant percept to the belief base together with the value of the sensor.

The following code snippet shows how the environment can make a connection to an MQTT broker and how messages published under a certain topic can be translated into Jason beliefs in the method `messageArrived` by means of the internal action `addPercept`.

```java
/* Called before the MAS execution with the args informed in .mas2j */
@Override
public void init(String[] args) {
    // [...]
    try {
        client = new MqttClient("tcp://localhost:1883", "JasonMQTTEnvironment");
        client.connect();
        // [...]
        client.setCallback(this);
```

```
          client.subscribe("MQTTJason/#"); // # is the wildcard for multiple topics
          // [...]
    }

  public void messageArrived(String topic, MqttMessage message) throws Exception {
    // Assuming a default format for publish messages:
    // MQTTJason/sensorid/temperature [for temperature]
    // MQTTJason/sensorid/humidity [for humidity]
    String[] components = topic.split("/");
    if ( components[2].equals("temperature") ) {
      addPercept(Literal.parseLiteral("temp_sensor(" +
                        components[1] +
                        "," +
                        new String(message.getPayload()) +
                        ")"));
          // [...]
         }
      }
  }
```

5.2.3 System Configuration

In Jason, it is possible to create a pre-specified number of identical types of agents using the same code, by using the multiplicity symbol #. The code snippet below creates a system running on a single processor with four kind of agents: the `controller`, `temp_sensor` (four instances), `humi_sensor` (eight instances), and one type of actuator, `airconditioner`.

```
MAS janus_test {
    infrastructure: Centralised
    environment: MQTTEnvironment
    agents:
         controller;
         temp_sensor #4;
         humi_sensor #8;
         airconditioner;
    classpath: "/Path/paho.client.mqttv3-1.0.2.jar";
    aslSourcePath: "src/asl";
}
```

5.2.3.1 The Individual Components

Humidity and temperature sensors. Both agents `humi_sensor` and `temp_sensor` implement similar behaviours: they broadcast the values for humidity and temperature respectively with a certain frequency. This is done either by the built-in `.broadcast(tell, humi_sensor(Me,H))` and `.broadcast(tell, temp_sensor(Me,T))` internal actions, or by bridging a value for the real world. In our case, we have used two DHT22 temperature and humidity sensors connected to two Raspberry Pi boards[1].

Air Conditioner. We use an Air Conditioner as the model of an agent that has to go through a predefined sequence of states. At the start it is in a waiting state. If it receives a message *turnon* from the controller, it will go through states *set_ up*, *cooling*, *cooldown*, and *waiting* again. The *turnon* message is only accepted if it is in the *waiting* state. Whenever there is a state variation, a message is broadcast so that it can be monitored by the controller.

Controller. The most complex part of the application logic resides in this agent. The controller continuously monitors the sensor values and does pattern matching of the individual components' behaviour, and prints the current state of the system according to a fidelity function. In particular, we implement three possible behavioural pattern functions for fidelity:

1. component values remain in a range,
2. values do not oscillate more frequently than X,
3. component goes through a sequence of states.

In the language of Section 4.2, Function 1 and Function 2 express sensor malfunction and thus ϕ_s, while Function 3 denotes ϕ_a, i.e. drifting of the actuator's value from the fidelity pattern[2]. Intuitively, the latter can be thought of as more akin to an "erroneous system deployment", since actuator `airconditioner` is the terminal interfacing the system's users.

Each function is implemented as a rule associated to each component. These individual rules are then used in the context of `controller`'s plans to raise a warning. For the temperature sensors, the first rule checks the Fréchet distance between the current value and a *baseline* value, defined as an affine periodic function on the hour of the day, so that the baseline ranges between 15 (midnight) and 25 (noon), with a valid range of baseline ± 5 degrees. For the humidity sensors, Function 2 is defined as too many requests in a given time window – for example a limit of 3 messages in 10 seconds. Finally, for the `airconditioner` agent, the rule checks whether it has gone through the correct sequence of states, thus something will be wrong if (a) there was no previous message, thus we are at the start, but we don't receive message "*waiting*"; or (b) there was an illegal transition, encoded by listing all the possible transitions in sequence.

[1] `https://www.adafruit.com/products/385`

[2] In [De Florio & Primiero 2015], where cyber-physical systems with users are under consideration, the patterns for Function 1 and Function 2 are used to express machine fidelity, Function 3 denotes user fidelity.

5.2.3.2 System Fidelity

In the setting of [De Florio & Primiero 2015], fidelity driftings are calculated based on system processes with variables of a given type that are associated with some shared volatile memory segments and read/write access rights. Since Jason runs on top of the Java virtual machine, we keep track of pattern violations directly in the `controller` agent code. Fidelity is then approximated for each pattern type as the inversely proportional function of the drifting in the reporting of raw facts, enabling a continuous cycle of monitoring, perception, control and action.

In our Jason evaluation, the fidelity function corresponding to each pattern is defined as a rule keeping track of violations and the dynamical evaluation of trustworthiness of the system happens in the main `controller` loop at run-time as relevant plans for each outcome are selected if certain patterns in the fidelity evaluation match as true. Each plan also defines an action that is executed in response to each evaluation, for example disabling a sensor or stopping the entire system.

In a system based on mechanical rules, the trustworthiness assessment produces four outcomes based on the model in Section 4.2:

1. *Trustworthy*, with high levels of fidelity on all functions, inducing optimal, sustainable working conditions;

2. *Unstable*, with high-to-medium values on Function 3, and low values on at least one of Function 1 or Function 2, inducing reconfigurable working conditions;

3. *Unsafe*, with high-to-medium values on both Function 1 and Function 2, and low levels on Function 3, inducing alarm-rising working conditions;

4. *Untrustworthy*, with low-levels of fidelity on all functions, inducing inadvisable or below-safety working conditions.

We present further tests and elaborations on the performance of the *Jason+Janus* client simulating the AC system described here in Subsection 6.1.2.

5.3 Reasoning about Contradictory Information

In this Section, we describe the implementation details behind the logical model (`un`)`SecureND`$^{\texttt{sim}}$ introduced in Section 4.3 and the experiments performed for the papers [Primiero *et al.* 2016] and [Primiero *et al.* 2017]. The following implementation and ensuing experimental analysis primarily concern:

1. changes in the final distribution of contradictory information according to network topology;

2. changes in the final distribution of contradictory information according to the ranking and epistemic role of seeding agents;

3. the quantification of the epistemic costs for trust and distrust operations.

5.3.1 Model Design

The model of (`un`)`SecureND`$^{\texttt{sim}}$ requires translating the proof system into portable procedures so that a implementation using any agent programming language is straightforward. In the case of the two published papers detailed in this section, all the programming is done in NetLogo with the `nw` extension. Recall from Section 3.7 that a network $\mathcal{G} = (V, E)$ is endowed with a set $V = (v_1, \dots, v_n)$ of vertices representing agents and a set $E = (e_{i,j}, \dots, e_{n,m})$ of undirected edges encoding information transmission. For each $v_i \in V$ let $v_i(p)$ and $v_j(\neg p)$ denote agents i and j knowing atomic formulas p, and $\neg p$, respectively; $v_k()$ denotes an agent k who knows nothing yet. Consequently, case $e(v_i(p), v_j())$ denotes a transmission channel from agent i to agent j such that the former can transmit p over to the latter, and case $e(v_i(p), v_j(\neg p))$ represents an admissible edge where two agents hold contradictory information, requiring a resolution procedure; finally, case $e(v_i(p), v_j(), v_k(\neg p))$ is shorthand for the triad case where one agent v_j does not hold any knowledge yet receives contradictory information from two distinct agents, i.e. there exist two edges between three nodes. Our NetLogo implementation preserves the order relation $\leq$ over $V \times V$ from (`un`)`SecureND`$^{\texttt{sim}}$, allowing for the possibility of a partial order in view of the different topologies; total networks where each agent has a connection to everyone else, and linear ones – where each agent is connected to an agent up in the hierarchy – have a total order, while scale-free networks, constructed using the Barabasi-Albert preferential attachment mechanism[3] and random networks – of Erdös-Renyi type – only have partial orders as some elements may not be comparable. We set the maximum number of vertices allowed at 300.

5.3.2 Implementation

The `Transmission` algorithm in Figure 5.5 describes how randomly seeded contradictory information $p, \neg p$ spreads across the network $\mathcal{G}$: for each edge between a node labelled by p and one with an empty label, if the latter node is sceptic and, in this case, if it is part of a predetermined 95% proportion of the sceptic population[4], or if its ranking is higher than that of the sender, it then calls the `Verify` procedure and it is added to the network with the new label p. Note that here we have implemented a random selection

[3] Where each unattached node creates an edge to other agents using a probability proportional to their existing neighbours.

[4] This sceptic proportion can be varied by the user in experimental setting by using a slider interface: indicatively, there are no substantial differences with choices set between 95 and 55.

```
PROCEDURE Transmission($G$)
$G := (V, E)$

FOR $e(v_i(\phi), v_j()) \in G$
IF (($v_j() \in$ sceptic) AND (random-float $1 =< 0.95$)) OR
    ranking($v_j()$) $<$ ranking($v_i(\phi)$)
THEN Verify($e(v_i(\phi), v_j())$) AND $G := G \cup (v_j(\phi))$
ELSEIF (($v_j() \in$ lazy) AND (random-float $1 =< 0.80$))
THEN Distrust($e(v_i(\phi), v_j())$) AND $G := G \cup (v_j(\neg\phi))$
ENDIFELSE
ENDFOR

FOR $e(v_i(\phi), v_j(), v_k(\neg\phi)) \in G$
SolveConflict($e(v_i(\phi), v_j(), v_k(\neg\phi))$)

RETURN Trusted($G$)
ENDPROCEDURE
```

Figure 5.5: Algorithm for Simple Information Transmission

of a 5% of sceptic agents who do not ask for verification; if the receiver is lazy and it is part of a predetermined 80% of the lazy population, it calls the Distrust routine and the new node is added with an opposite label (again, we have here set a random 20% of sceptic agents who do not distrust the information).

Thus, the epistemic description of our agents follows the basic distinction between lazy and sceptic agents of Subsection 4.3.2, but to provide a more realistic description of our MAS, we allow for random changes of attitude, and this is reflected in the proportions of sceptic and lazy agents in the network being sufficiently lower than 100. In particular, we allow a very low rate of verification cases for lazy agents in networks that have a majority of such agents; and a similar rate for sceptic agents that might accept information without implementing verification in networks that have a balanced distribution of lazy and sceptic agents. The MAS design implements three fixed distributions of this epistemic attitude across the networks coupled with a semi-random implementation of the corresponding procedures, summarised into three configurations of networks with different proportions of sceptics and their confirmation rates:

- ▷ *overly lazy network*: the proportion of sceptic nodes is set at 20%, with their confirmation rate at 5%, the latter expressing the proportion of such agents that will after all ask for verification;

```
PROCEDURE Verify($e(v_i(\phi), v_j())$)

set COSTTRUST+1
set TRUSTLINK $e(v_i(\phi), v_j(\phi))$
RETURN Trusted($G$)
ENDPROCEDURE
```

Figure 5.6: Algorithm for Trust Costs Increase

- ▷ *balanced network*: the proportion of sceptic nodes is set at 50% and their confirmation rate at 95%, to account for a 5% of random sceptic agents who decide not to ask for verification after all;
- ▷ *overly sceptic network*: the proportion of sceptic nodes is set at 80%, their confirmation rate at 100%, always implementing verification.

The rationale is as follows: we assume that in a network with a large majority of agents with virtuous behaviour, this is preserved; whereas in a balanced network, we allow a low number of virtuous agents to slip in their habits; finally, in a network largely characterised by lazy behaviour, we still allow some of the agents to be influenced by the few ones that have a sceptic attitude.

The procedure `Verify` is shown in Figure 5.6: its role is to increase the value of costs associated with the number of *trusted links*. A successful confirmation procedure establishes trust as a property characterising edges and it equals 0 at setup/initialisation stage. Accepting information means in turn creating a trusted edge – marked green in the simulation in Section 6.2.2 – and to acquire knowledge of the atom passed. A graph G is relabelled to `Trusted(G)` by the procedure `Transmission`. Notice that passages where the receiver agent is lazy or lower in the dominance relation do not generate a trusted link. The subroutine `Distrust` is illustrated in Figure 5.7: it increases the value of costs associated with the number of distrusted links and it labels the receiving node with the contrary atom to the one received – i.e. it applies a negation.

When a node labelled by an atom p (by a previous interaction) is linked to another node labelled by the contradictory $\neg p$, the routine `SolveConflict` is started. Two versions of this resolution strategy are presented below.

The first version, detailed in see Figure 5.8, takes into account the number of links with nodes labelled by p, the number of links with nodes labelled by $\neg p$ and sums them to the respective overall rankings, obtaining values `ScoreP` and `Score¬P`. This implementation sensibly refines the pure majority by counting of the formal system in Section 4.3.2, by adding the ranking of the agents involved as a parameter of the related score. For each pair of edges from nodes with contradictory information $p, \neg p$ to

```
PROCEDURE Distrust($e(v_i(\phi), v_j()))$

set COSTDISTRUST+1
set DISTRUSTLINK $e(v_i(\phi), v_j(\neg\phi))$
RETURN Trusted($G$)
ENDPROCEDURE
```

Figure 5.7: Algorithm for Distrust Costs Increase

```
PROCEDURE SolveConflict($e(v_i(\phi), v_j(), v_k(\neg\phi)))$

TotalP $= \#(V_i(\phi), V_j())$
Total¬ P $= \#(V_k(\neg\phi), V_j())$
ScoreP $= 1/$ranking$(V_i(\phi)) + ($TotalP$/\#V)$
Score$\neg P = 1/$ranking$(V_i(\neg\phi)) + ($Total¬ P$/\#V)$

FOR $e(v_i(\phi), v_j(), v_k(\neg\phi)) \in$ Transmission($G$)
  IF (ScoreP $>$ Score $\neg P$)
    THEN $G := G \cup v_k(\phi)$
  ELSEIF (ScoreP = Score$\neg P$)
    THEN $G := G \cup v_k(\neg\phi)$
  IF (random-float $1 >= 0.5$)
    THEN $(v_k(\phi))$
    ELSE $(v_k(\neg\phi))$
  ENDIF
ENDFOR

RETURN Trusted($G$)
ENDPROCEDURE
```

Figure 5.8: Algorithm for Conflict Resolution by Trust Majorisation

an unlabelled node, if the value of ScoreP is higher than the value of Score¬P, the new node is labelled by p, otherwise by $\neg p$. We assume here a context in which agents refer to a popularity criterion of *most trusted* in order to choose which of two contradictory pieces of information to preserve.

The second version, shown in Figure 5.9, analyses the number of distrusted links appended to each neighbour with each contradictory piece of information and selects

```
PROCEDURE SolveConflict2($e(v_i(\phi), v_j(), v_k(\neg\phi))$)

let d1 #DISTRUSTLINK $e(v_{i,\dots n}(\phi), v_j())$
let d2 #DISTRUSTLINK $e(v_{k,\dots m}(\neg\phi), v_j())$

IF (length d1 > length d2)
  THEN $G := G \cup (v_j(\neg\phi))$ AND Distrust($e(v_i(\phi), v_j(\neg\phi))$)
ENDIF

IF (length d1 < length d2)
   THEN  $G := G \cup (v_j(\phi))$ AND Distrust($e(v_k(\neg\phi), v_j(\phi))$)
ENDIF

IF (length d1 = length d2)
  IF (random-float $1 =< 0.5$)
  THEN $G' := G \cup (v_j(\neg\phi))$ AND Distrust($e(v_i(\phi), v_j(\neg\phi))$)
  ELSE $G' := G \cup (v_j(\phi))$ AND Distrust($e(v_k(\neg\phi), v_j(\phi))$)
  ENDIFELSE
ENDIF
ENDPROCEDURE
```

Figure 5.9: Algorithm for Conflict Resolution by Distrust Minorisation

the new label from the least distrusted one, proceeding by random choice when an equal number of distrusted links is detected. Here a context is assumed in which agents refer to a popularity criterion of *least distrusted* in order to choose which of two contradictory pieces of information not to preserve. It then executes the procedure `Distrust` on the selected link.

The observer-property of *trustworthiness* – i.e., the total number of trusted links – and *distrustfulness* – i.e., the total number of distrusted links – first defined in Subsection 4.3.3 are at any given time a known property of the network and are used to perform conflict resolution. The procedure ensures that `clearP`, trusted and distrusted links obtained by a first message passing operation are preserved in subsequent executions of the procedure `Transmission` over the same graph to analyse their effect on epistemic costs.

5.4 Simulating DAG-based Distributed Ledgers

In this Section, we describe the implementation of the Tangle DAG simulation environment introduced in Section 4.4. It is clear that after the genesis transaction, the growth of $\mathcal{T}$ is uniquely described by the attachment rules, i.e. the tip selection mechanism. The behaviour of this process in the long run is akin to a random graph in a random environment and, like similar models such as diffusion-limited aggregation [Wikipedia 2018b] is simple to state but difficult to analyse mathematically; however a simulation and a visual representation aid considerably in eliciting its structure and properties. Our models and analyses of the growth of the Tangle are implemented entirely in NetLogo; code and examples are freely available at `https://bitbucket.org/mdxmase/iotasim/`. We also build a strategy selector menu which can be used to implement different strategies in function of other node internal variables in addition to the cumulative weight.

5.4.1 Random Uniform Growth

The simplest version of our NetLogo code, based on the random growth of Subsection 4.4.4 with instantaneous approvals, makes it easy to generate fast, efficient samples of the Tangle. For example, one can easily scale and visualise in real time up to tens of thousands of nodes being added and confirmed; on a machine with 16GB of RAM and a 2.7GHz multicore i5 processor running MacOS 10.13 or Ubuntu 17.10 this takes about 10 minutes. We define two initial global parameters, `genesis` for the genesis transaction(s), and `lambda` for the average number of incoming tips, which is assumed to be drawn from a Poisson distribution via a reporter function. We also assume that turtles (called nodes by the use of a `breed` keyword) have an internal variable *cw* that records their cumulative weight. Likewise, directed links (breeded as edges) are used to construct the Tangle structure. Other internal variables are possible and commented in the more complicated models. The procedure `setup-tangle` initialises the Tangle by creating a star network of edges directed from the initial tips to the genesis, and sets their initial *cw* which, for simplicity, is 2 for the genesis and 1 for the tips. Thereafter, the procedure `grow-tangle` does three things, implemented as further procedure calls: a) it finds tips to attach – i.e. it approves nodes created by the Poisson clock; b) it updates the internal variable *cw* by incrementing it if a node has an incoming edge; and c) it advances the simulation by one discrete tick. Optionally, it can also update the spatial layout and any visual feedback before the `tick` command is given, for example by colouring nodes with similar *cw*. Both main procedures above, and the ancillary `approve-nodes`, `update-cw` procedures and `incoming-tips` reporter, take less than 20 lines of terse code, visible in the snippets of Figure 5.12. The typical shape of a large random Tangle sample and of its adjacency matrix is shown in Figures 5.10a and 5.10b; its depth is described by the layering of cumulative weight, and its width by the average incoming tips λ, which is user-modifiable.

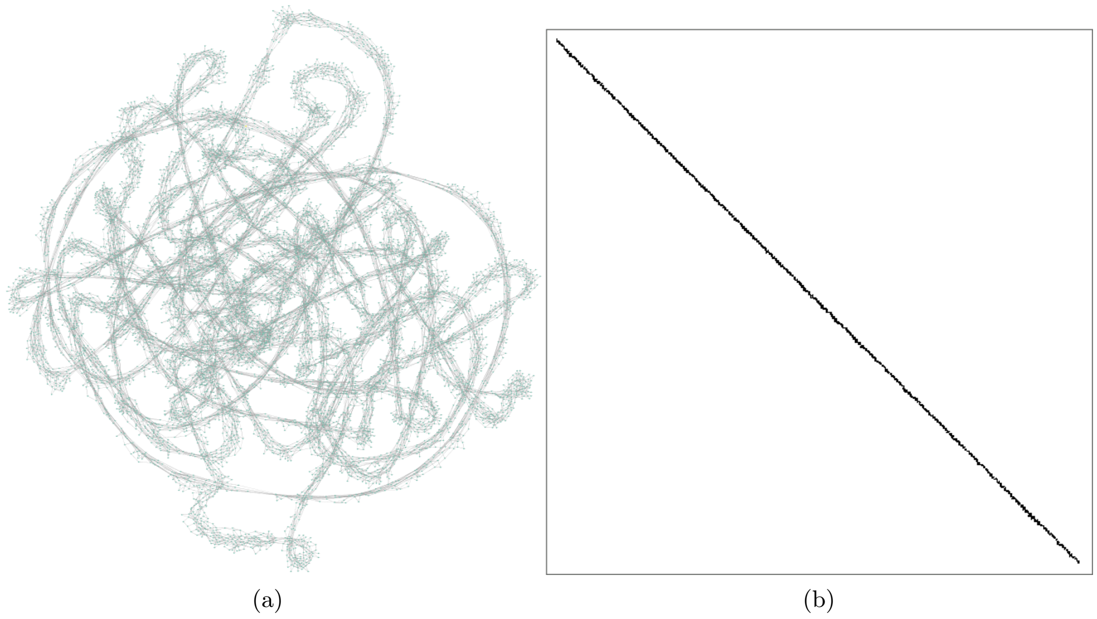


Figure 5.10: The "typical" shape of a random Tangle at $\mathcal{T}_{1010}$, with 1039 nodes, $\rho = 3$, $\lambda = 10$, and its adjacency matrix $\mathcal{A}$.

To highlight its information accumulation nature, we also implemented a simple "colour history" of the Tangle by adding two lines of code in the `grow-tangle` procedure to change the node colour once its internal variable *cw* has been updated. NetLogo currently supports a simple 0-139 fractional scale from predefined discrete colour swatches; in Figure 5.11 colour hues cycle through this scale[5].

5.4.2 Growth Using the MCMC Selection Algorithm

We replicated the Tangle growth strategy using the MCMC-type algorithm, which we simplified for computational tractability by experimenting with the array and network extension primitives `nw:turtles-in-radius`, `nw:path-to`, `nw:turtles-in-reverse-radius` and recursive `ask one-of in-edge-neighbors` calls until we found a satisfactory solution, and packaged it into a `rw-tips` approval procedure that for each random list of new tips created by the Poisson clock, initialises the chains based on a `num-walkers` parameter and manually backpropagates the Tangle edges to the two selected tips. This is less efficient than random uniform choice of tips, but speeds the computations by a factor of 10 compared to an algorithm that searches the state space at every tick via the transition matrix and a list copy, which is

[5]It is also possible to use a more complex RGB/RGBA colour rendering, as in **set color [245 231 42]**, by specifying a list of integers to increase.

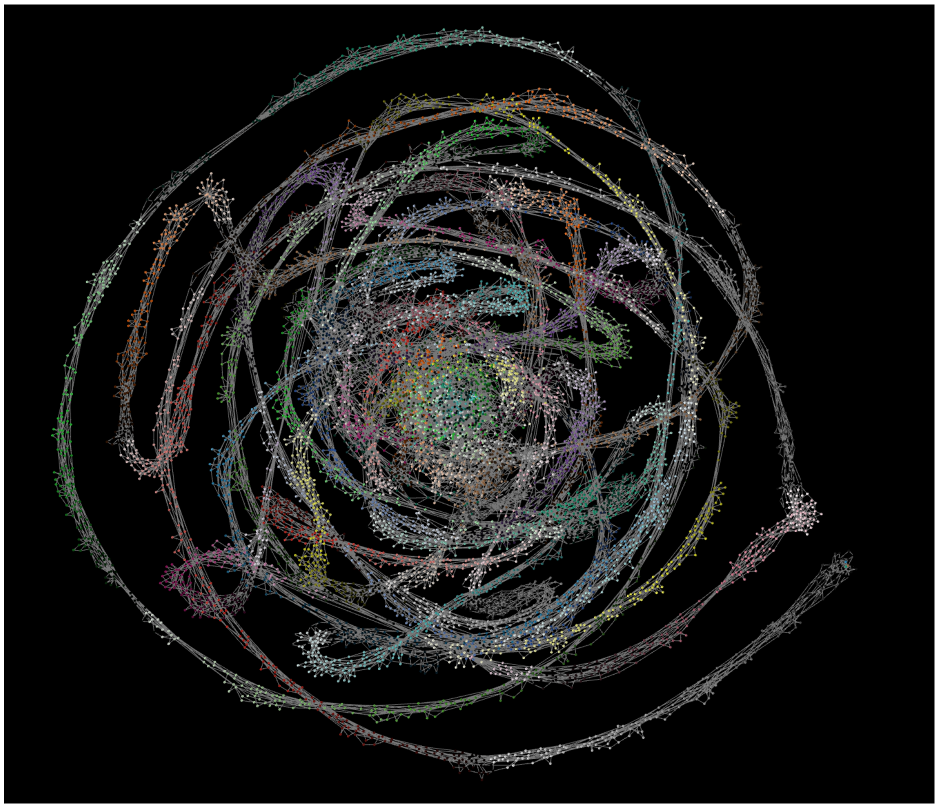
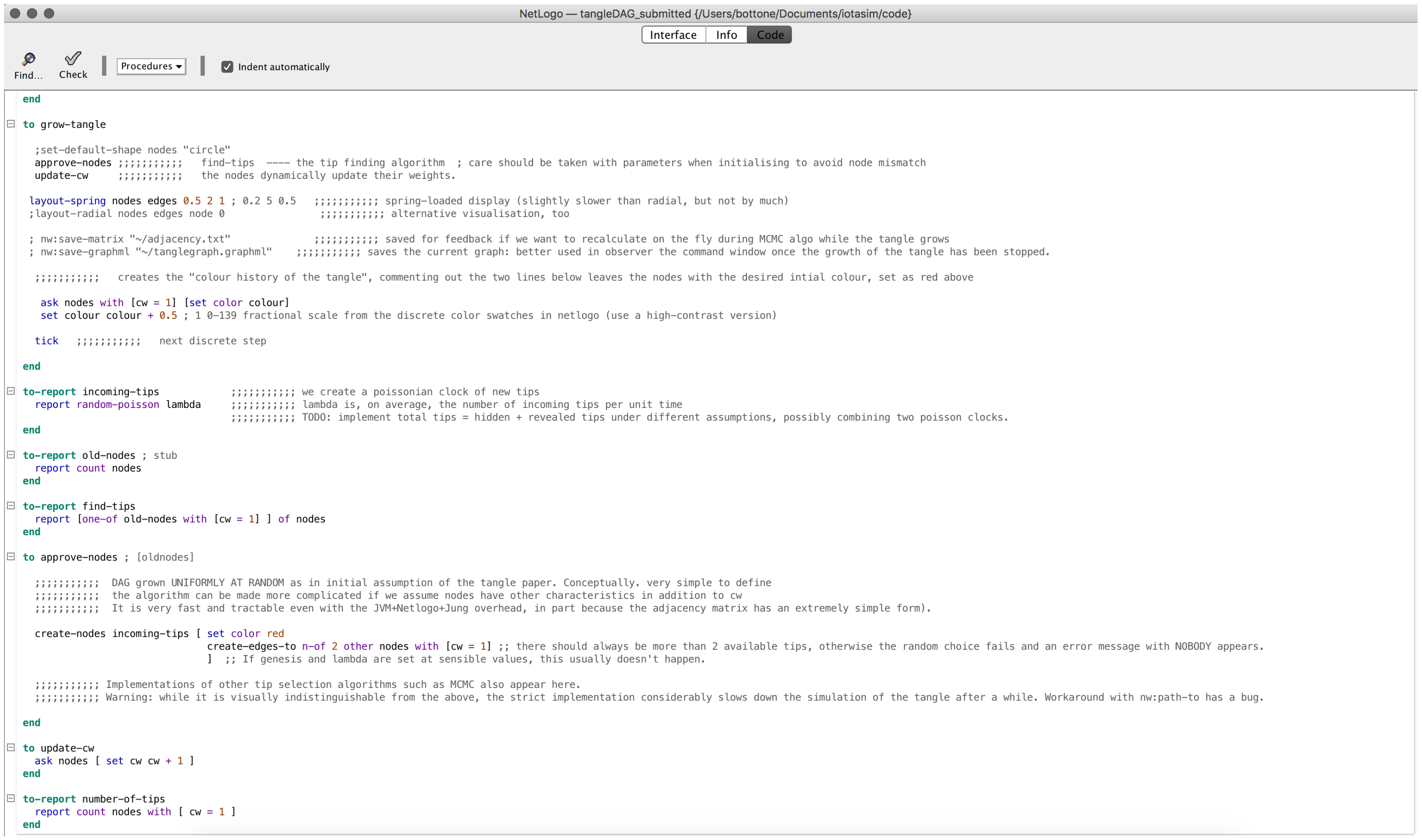


Figure 5.11: NetLogo rendering of colour-coded layers of the *cw* variable

computationally expensive for a large (≥ 100) number of walkers.


NetLogo — tangleDAG_submitted {/Users/bottone/Documents/iotasim/code}
Interface
Info
Code
Find...
Check
Procedures
Indent automatically

```
end

to grow-tangle

  ;set-default-shape nodes "circle"
  approve-nodes ;;;;;;;;;;;   find-tips  ---- the tip finding algorithm  ; care should be taken with parameters when initialising to avoid node mismatch
  update-cw     ;;;;;;;;;;;   the nodes dynamically update their weights.

 layout-spring nodes edges 0.5 2 1 ; 0.2 5 0.5   ;;;;;;;;;;; spring-loaded display (slightly slower than radial, but not by much)
 ;layout-radial nodes edges node 0                ;;;;;;;;;;; alternative visualisation, too

 ; nw:save-matrix "~/adjacency.txt"              ;;;;;;;;;;; saved for feedback if we want to recalculate on the fly during MCMC algo while the tangle grows
 ; nw:save-graphml "~/tanglegraph.graphml"    ;;;;;;;;;;; saves the current graph: better used in observer the command window once the growth of the tangle has been stopped.

  ;;;;;;;;;;;   creates the "colour history of the tangle", commenting out the two lines below leaves the nodes with the desired intial colour, set as red above

   ask nodes with [cw = 1] [set color colour]
   set colour colour + 0.5 ; 1 0-139 fractional scale from the discrete color swatches in netlogo (use a high-contrast version)

  tick   ;;;;;;;;;;;   next discrete step

end

to-report incoming-tips            ;;;;;;;;;;; we create a poissonian clock of new tips
  report random-poisson lambda     ;;;;;;;;;;; lambda is, on average, the number of incoming tips per unit time
                                   ;;;;;;;;;;; TODO: implement total tips = hidden + revealed tips under different assumptions, possibly combining two poisson clocks.
end

to-report old-nodes ; stub
  report count nodes
end

to-report find-tips
  report [one-of old-nodes with [cw = 1] ] of nodes
end

to approve-nodes ; [oldnodes]

  ;;;;;;;;;;;  DAG grown UNIFORMLY AT RANDOM as in initial assumption of the tangle paper. Conceptually. very simple to define
  ;;;;;;;;;;;  the algorithm can be made more complicated if we assume nodes have other characteristics in addition to cw
  ;;;;;;;;;;;  It is very fast and tractable even with the JVM+Netlogo+Jung overhead, in part because the adjacency matrix has an extremely simple form).

  create-nodes incoming-tips [ set color red
                               create-edges-to n-of 2 other nodes with [cw = 1] ;; there should always be more than 2 available tips, otherwise the random choice fails and an error message with NOBODY appears.
                               ]  ;; If genesis and lambda are set at sensible values, this usually doesn't happen.

  ;;;;;;;;;;; Implementations of other tip selection algorithms such as MCMC also appear here.
  ;;;;;;;;;;; Warning: while it is visually indistinguishable from the above, the strict implementation considerably slows down the simulation of the tangle after a while. Workaround with nw:path-to has a bug.

end

to update-cw
  ask nodes [ set cw cw + 1 ]
end

to-report number-of-tips
  report count nodes with [ cw = 1 ]
end
```

Figure 5.12: NetLogo Tangle DAG simulator code view

Chapter 6

Evaluation

In this Chapter, we give a reasoned evaluation of the results corresponding to the underlying models presented in Chapter 4 and their implementation as described in Chapter 5, and discuss the computational costs of the simulations and their relevance to real-world performance, whereas the overall evaluation with respect to the objectives of this thesis can be found in Chapter 7.

6.1 Evaluation of MQTT-Jason Bridges and AC Monitor

In this Section, we offer considerations and evaluations for the experiments based on our bridging solutions discussed in Sections 4.1, 4.2, 5.1 and 5.2.

6.1.1 System Evaluation

It should be noted that the multi-agent systems implemented for the papers [Bottone *et al.* 2016a] and [Bottone *et al.* 2016b], and in particular [Bottone *et al.* 2016c] share a common architecture based on Java bindings to the chosen MAS simulation environment (in this case, Brahms and Jason). Ultimately, when setting up the bridges between MQTT and the multi-agent system, once the helper methods and the appropriate interface to the messaging protocol and client/server callbacks are implemented for one application, it is relatively straightforward to modify and extend this framework to test it using a different protocol and broker, and the performance and results of the simulation only vary in function on the number of agents modelling each component and the computational power of the CPU and RAM bandwith used in experiments, as each agent makes a system call for each percept. In particular, the systems developed in Section 5.1 have been tested over a typical load of dozens of robots, in the case of the stigmergic approach, while the one described in Section 5.2 has involved a setup consisting of hundreds of components.

We have found that, as a rule, system performance scales well with the number of agents up to a few hundreds (500 being a typical example), by which point the dual computational overhead given by the Jason/Brahms BDI implementations starts to bite into the percept messaging callbacks and slows down the system. Nevertheless, for the type of real world systems envisaged in these experiments, the performance and computational cost expended for a system running on a single machine behaves as expected, with the lightness of MQTT and the informational overload feature reduction given by the particular model implemented with the publish/subscribe topic segmentation efficiently parcelling information flow within the system.

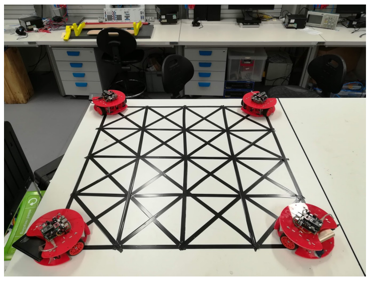
The model given in [Bottone *et al.* 2016b] has proved to be quite robust to the progressive increase in messages vis-a-vis the mapping to four trustworthiness evaluations, even with the simple rotation of states described in Subsubsection 5.2.3.1. This is a key feature we observed in practice. Note that, unlike the setting of the experiments performed in the ensuing Sections, the multi-agent systems studied here adopt a *Centralised* scheduler overseen by the MAS environment built using Brahms and Jason, rather that a truly distributed architecture based on a network topology. For this reason, performance bottlenecks are entirely ascribable to the inter-process communication of the centralised MAS system, and performance using, say, a decentralised agent communications protocol such as FIPA/JADE (discussed in [Bordini *et al.* 2007]) has not been investigated and has been left for future extensions.

6.1.2 Performance

We present below a quantitative evaluation of the performance of two representative agoric systems that we implemented; we repeat here the experimental setups for clarity.

Stigmergy

The stigmergic system in [Bottone *et al.* 2016a] consisted of two parallel simulations as described in Subsection 5.1.3: a virtual one using the Jason/BDI Java frameworks running in Eclipse, performed on a Macbook Pro 2012 with 16GB of RAM, and a 2D graphical environment as in Figure 5.4; and a second, more limited live experiment consisting of four robots incorporating the MIRTO robotic platform [Androutsopoulos *et al.* 2018], moving on a scaled down 5×5 square enclosure where each position in the grid contains exactly one robot and robots change positions by following black lines arranged in a triangular Petersen mesh, shown in Figure 6.1(left). The Mirto robot, with two motors, infrared sensors, geared motors with wheels, bump contact sensor, potentiometer, buzzer and LCD screen connected to an Teensy Arduino microcontroller and a Raspberry Pi 3 and support for other sensors and actuators, is well suited as an inexpensive and resource-limited, yet expressive, capable and expandable system that can be used to interact with its environment. Unlike the Jason simulation setting, the MIRTOs have to implement an inter-agent communications protocol on their Raspberry Pi boards for communicating to the MQTT broker and a mechanism for reading environmental variables. This was done in the following way: a map of the mesh environment is coded on a coordinating back-end server on a laptop to which the robots are connected on a wireless network, and robots are only allowed a fixed distance from their position for each step in the simulation, corresponding to grid positions on which the virtual pheromone map is broadcast: the local pheromone readings in the form $(\mathbf{x}, \mathbf{y}, \mathbf{i})$ where the nine-cell marks and the intensity $\mathbf{i}$ are in the form of MQTT messages which are processed by custom code on the Raspberry Pi and sent back to the laptop. Initially robots are at the four corners and continuously read their position and compare it to the pheromone map. If they bump into the enclosure walls or another robot, they register that direction as inaccessible and turn at random until they find a line, and try to advance a fixed distance, then query the laptop for a pheromone trace, until they find one, and update the global pheromone map according to the procedure described in Subsection 5.1.3. In this scaled-down experiment, we observed that the robots exhibit flocking behaviour. The results of the Jason MAS experiment are shown in the right Table in Figure 6.1; for each instance of Jason we ran a batch of 100 simulations of the MAS, and recorded the computational cost of the simulations in terms of three components: (a) the number of agents, (b) the number of MQTT broker messages per minute (MPM), and the memory cost in megabytes (MCM), averaged over the simulation run[1]; We use these two markers to provide an estimate of the performance of the system as its complexity increases; we note that for a large part the performance of the system is constrained by the size of its virtual environment, given by the global pheromone map, as the number of robots increases.



♯Agents	Average ♯ MPM	Average MCM
10	33.17	56MB
50	169.43	311MB
100	358.91	705MB

Figure 6.1: Left: The live environment for the MIRTO experiment. Right: Table of computational costs for the Jason simulation from [Bottone *et al.* 2016a]

AC System

We tested the performance of the *Janus+Jason* client by instantiating up to several hundreds temperature and humidity sensors. Running times were acceptable up to a system size of 500 components, when the overhead of the Java virtual machine in Jason becomes too much to handle for a quad-core Retina MacBook Pro 2014 with 8GB of RAM. Given that real-life smart home systems tend to be in the dozens of components and large systems – such as those of a skyscraper – are composed of thousands of sensors and actuators, initial evaluation has been deemed encouraging.

Table 6.2 gives a bird's eye overview of the average cost of the simulations for the MAS corresponding to the paper [Bottone *et al.* 2016b], described in Section 5.2.1 in terms of the same markers as above, namely the number of agents/components, MPM, MCM and an additional one given the cumulative number of callbacks to state change (CSC), which is used to roughly reflect the proportion of times that the system does not pass in a *waiting* state, normalised for the overall number of states (4 in this case). We recall that in our setup, the Air Conditioner is modelled as a standalone

[1]The simulation was run on Eclipse Luna for MacOS, where the relevant process is sandboxed.

agent to which the remaining $N - 1$ sensors broadcast their values – either simulated or bridged from the DHT22 sensor readings, equally subdivided between temperature and humidity sensors. As before, the sandboxing ensures that it straightforward to measure the requirements of the simulation. We observe that across al performance measures, the system requirements increase slightly higher than the expected linearity in the number of components, but remain in the linear class within the simulation range.

♯Agents	Average ♯ MPM	Average MCM	Average CSC
11	47.51	71MB	328
51	190.02	408MB	864.76
101	422.68	893MB	2451.98
251	1586.47	1761MB	6620.59
501	2287.51	3934MB	18434.02

Figure 6.2: Computational costs for the AC system used in [Bottone *et al.* 2016b]

6.2 Epistemic Costs of Trust

In this section we present experimental results and an evaluation of the simulations performed for the model of Sections 4.3 and 5.3. We ran the experiments over the four different types of networks. Unlike random networks, *scale-free networks* better represent the topology of graphs occurring in complex systems such as large social networks. On the other hand, *linear networks* are more common in hierarchical structures encountered in conditions of user access control where there exists a hierarchy of permissions. The experiments in this section have been executed on a machine with 7.7 GB of RAM running 64bit Ubuntu 15.10. Data from several scale-free networks of fixed dimensions between 10 and 300 nodes were systematically collected. The seeding of contradictory information is done by associating an atom p to a lazy node, and its negation to a sceptic one, although this association can be altered at will. The code and result of the experiments are available at `https://bitbucket.org/bottone/securendsim`.

A first set of experiments has been done on what we call a *memoryless* network, meaning that for each successive run of the algorithm, the structure of the graph is re-plotted. As a result, all previous trusted edges are forgotten, meaning that the next information distribution is not affected by the previously established links. For these network types, we have run a total of 240 executions of the main algorithm, subdivided into 30 runs for each size of $10, 20, 30, 40, 50, 100, 200, 300$ nodes. The following results can be consistently observed across experiments:

- ▷ The knowledge plot, i.e. the final labelled graph, is never consistent with the previous execution;
- ▷ There is no systematic distribution of consensus across the 30 runs;
- ▷ There is no systematic relation between the resulting knowledge plot and the costs of the information transmission;
- ▷ There is no systematic relation between the knowledge plot and the ranking of the initially seeded nodes (i.e. the labels at the beginning).

The above findings indicates that memoryless networks do not offer a reliable experimental setting to investigate issues of consensus and epistemic costs of trusted graphs.

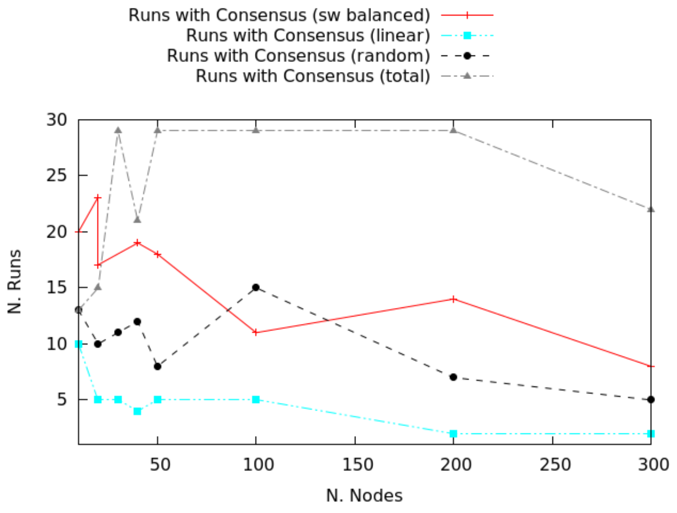
The second set of experiments was performed again on several networks of different size, but ensuring at each run of the main algorithm that the trust graph obtained during a previous run is preserved. This is obtained by executing at the end of each run a procedure called `clearP` (detailled in Subsection 5.3.2) that eliminates all labels from the graph yet preserves ranking and trusted links. In this way we can better average on the number of trusted and distrusted links which are created and destroyed during each execution. Under these experimental conditions, we then analyse consensus (Subsection 6.2.1), epistemic costs (Subsection 6.2.2), ranking of the seeding nodes (Subsection 6.2.3), epistemic attitude (Subsection 6.2.4) and time complexity (Subsection 6.2.5) in networks with trust and distrust.

6.2.1 Reaching Consensus

A graph that satisfies consensus is called a *unanimous graph*. Network configuration directly affects consensus results in memory-preserving networks with trust only, see top graph of Figure 6.3, summarised as:

- ▷ Total networks reach consensus most often;
- ▷ Scale-free networks always perform better than linear or random ones in terms of number of runs that reach consensus;
- ▷ The data for random networks is not overly reliable, as full labelling might not be reached (proportionally often in the number of nodes); For this reason, a timeout is set at 1000 steps[2]. The proportion of runs that timeout is given in the bottom graph of Figure 6.3, showing a non-strictly linear increase. Accordingly, the number of runs that reach consensus is bound to decrease.

[2]A step indicates here one message passing operation.



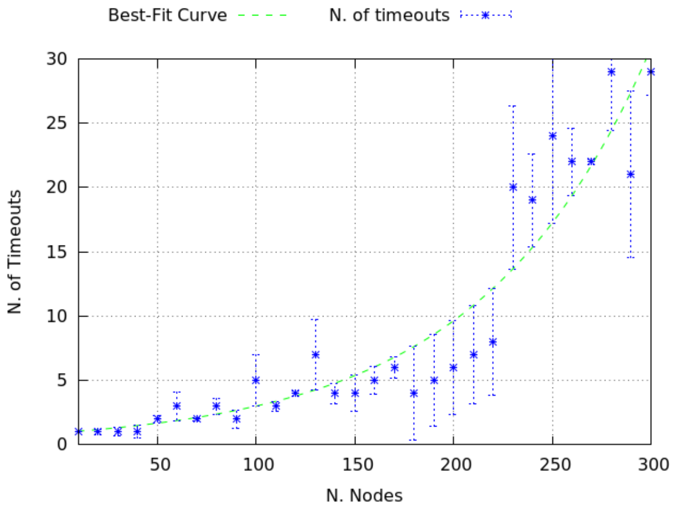


Figure 6.3: Top: Comparison of consensus between networks with trust. Bottom: Proportion of timeouts in random networks.

In particular, for scale-free networks (labelled as `sw` in the Figures that follow) the clustering of lazy nodes is inversely proportional to the construction of trusted edges and in larger networks disadvantageous to consensus reaching transmissions. Increasing the number of nodes while keeping the `sceptic-lazy` proportion unchanged, we are also bound to increase the probability of clusters of lazy nodes, thus to reduce the number of trusted edges, in turn progressively reducing the number of runs in which consensus is obtained. Results for the three configurations – i.e lazy, balanced and sceptic scale-free networks – are plotted in Figure 6.4. These results show that in smaller networks (10 to 30 nodes), clusters of lazy nodes occur relatively often: the results for the three configurations are grouped in a restricted area, between 13 and 23 runs with consensus over 30. In these small networks, the denser groups of lazy nodes balance the reduced number of trusted links by sceptics. Once network size increases, the positioning of lazy nodes becomes more sparse. This topological factor crucially influences the number of runs in which consensus is reached: in general, overly sceptic networks perform better at becoming unanimous graphs. This can also be interpreted by saying that lazy nodes are less strict in preserving their labelling, i.e more prone to change their minds.

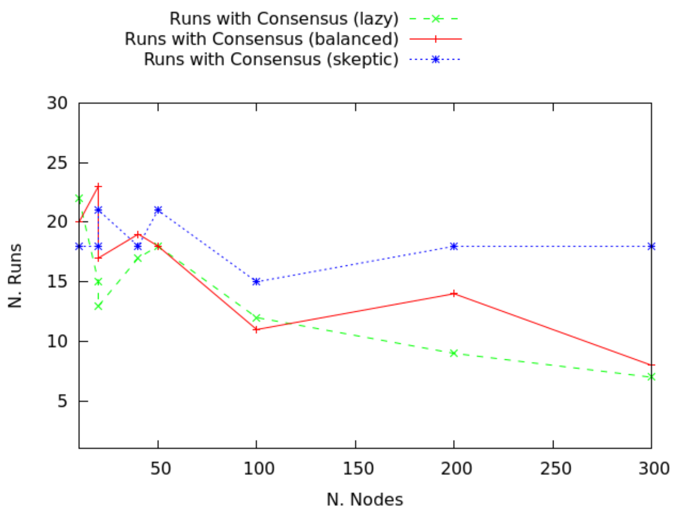
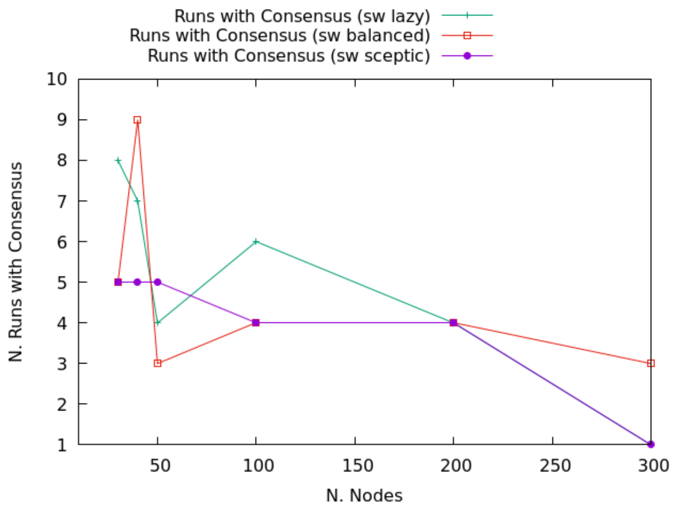


Figure 6.4: Consensus in scale-free networks

As shown in the top graph of Figure 6.5, networks with trust and distrust present an inverse correlation between size and the number of transmissions that reach consensus: the smaller the network, the more often full labelling with a unique formula is obtained (i.e., it is easier to reach consensus). Despite some differences in the reached peaks by lazy and balanced networks, the overall behaviour is similar in all configurations: balanced networks have the highest absolute number of such runs, while networks with higher proportion of sceptic agents have the lowest number of consensus reaching transmissions. Networks with distrust significantly differ from those with trust only for the total amount of consensus-reaching transmissions. We show this for balanced networks in the bottom graph of Figure 6.5, as the same holds for lazy and sceptic networks: the presence of a distrust routine has a strong impact on the ability of the network to reach consensus in the presence of contradictory information, with no more than 9% of runs reaching a full labelling by either p or $\neg p$ (network of 40 nodes), while in the case of networks with trust only this value reaches 63% (for networks of the same size).

The experimental results on consensus shown above empirically support the properties of (un)SecureND$^{\mathtt{sim}}$ derivations provided in Subsection 4.3.2. To observe this,



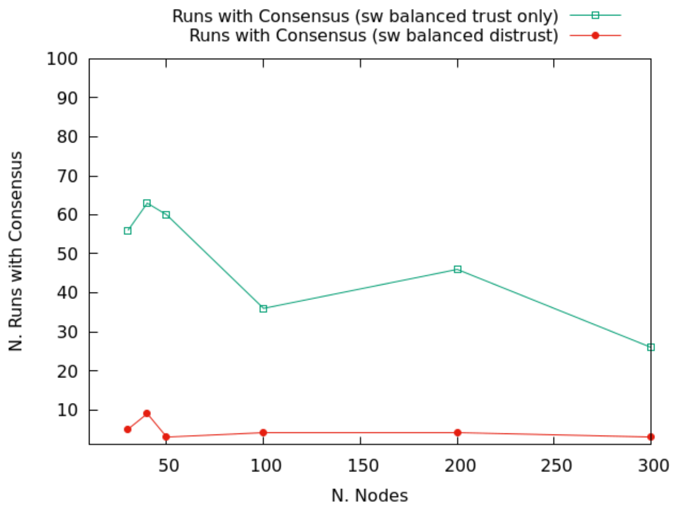


Figure 6.5: Consensus in scale-free networks with distrust. Top: for all lazy, balanced and sceptic configurations. Bottom: for balanced networks only.

consider that total networks are graphs in which the number of edges between nodes is maximal, corresponding to derivations with the maximal number of branches, one for each pair of agents (v_i, v_j) appearing respectively in the premises and in the conclusion. Similarly, overly sceptic networks are graphs corresponding to derivations where more instances of the `verify_sceptic` rule are used. In both cases, the number of executions resulting in consensus are maximal. On the other hand, linear networks are graphs corresponding to derivations where the number of agents for which the ranking can be transitively established is maximal, and overly lazy networks are graphs corresponding to derivations where more agents implement the `unverified_lazy` rule.

6.2.2 Epistemic Costs

The second type of experimental analyses we performed concerns *epistemic costs*. This term refers to the computational expenses required to perform verification and distrust operations: these correspond in the natural deduction calculus to instances of rules `verify_high` and `verify_sceptic` for trust and rules `unverified_contra` and `unverified_lazy` for distrust; in the algorithm they correspond to the `Verify` and `Distrust` procedures. The effect of these procedures in the network is to generate trusted and distrusted links, respectively. Given there are proportionally more nodes than links and a message might pass more than once over a given node through several senders, the values for costs are expected to be higher than those for links. In addition, given the conditions for `Verify` are more than those for `Distrust`, the values of trust can be expected to be higher than those for distrust. The aim is to assess these values in the different topologies, to evaluate the proportion between trust and distrust costs and to use them as parameters to evaluate these actions with respect to consensus and complexity. We start by comparing balanced scale-free with random and linear networks with trust only, as in the top graph of Figure 6.6. The results can be summarised as follows:

- ▷ Random networks are by far the most epistemically expensive;
- ▷ Linear networks are slightly less expensive than scale-free ones, by a very small margin. If one is balancing costs against information diffusion, scale-free networks should always be preferred to linear ones, since the associated costs do not diverge much[3];
- ▷ Given the previous observation on consensus and timeouts, it is obvious that random networks are the worst performing.

We now compare in more detail average trust costs in scale-free networks in all agent configurations (balanced, overly lazy and overly sceptic). The results are plotted in the bottom graph of Figure 6.6. By definition, a network with a higher number of sceptic nodes and confirmation requests will have higher trust costs. For small networks up to 40 nodes the costs are within a small range between 9 and 52; the difference increases significantly between larger overly lazy and overly sceptic networks. Cost difference remains comparably restricted for balanced and overly lazy networks (with a minimum difference of 15 average points at 50 nodes). This suggests that if one is trying to balance trust costs against consensus, large lazy networks should be preferred over balanced ones, as in the latter case the number of runs with uniform labelling tends to drop, while the costs still increase.

[3]As in general, scale-free networks are more resilient than linear ones.

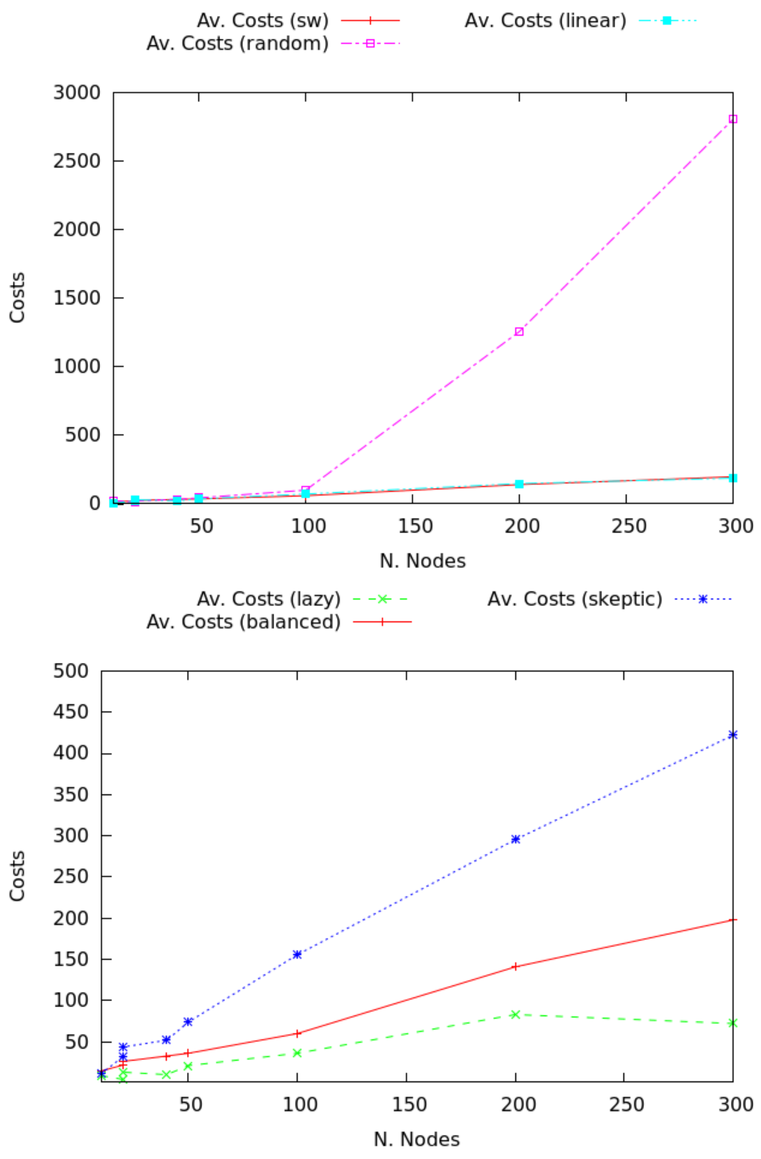
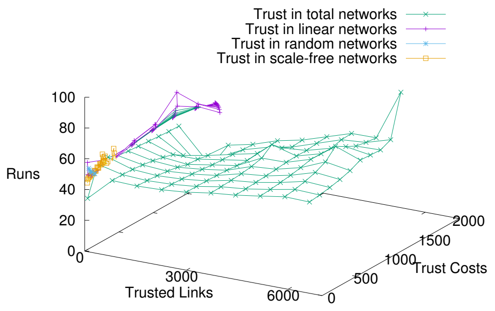


Figure 6.6: Top: Comparison of average costs of trust. Bottom: Average costs of trust in scale-free networks.

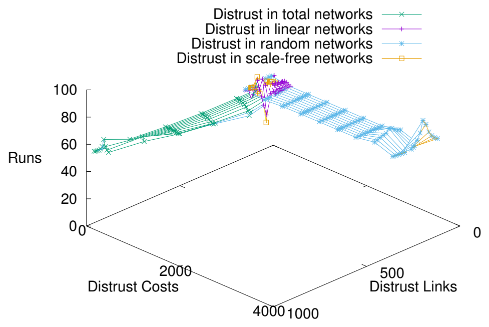
Figure 6.7 and the associated Table show that the average rate of links and trust costs is inversely proportional: the former increases from random, through linear, scale-free and total networks, while the latter decreases. Given the fixed number of sceptic agents across the various topologies, the decrease in costs should be mainly associated with the ranking of agents and their order, while the increase in trusted links is purely due to the number of links in the network. From these data it appears that random networks perform the worst, as the required costs are high but the obtained links are less than in scale-free or linear networks.



	Average Trusted Links	Average Trust Cost
Random	47.78	940.92
Linear	77.78	434.16
Scale-free	102.21	415.42
Total	971.44	385.11

Figure 6.7: Trust distribution and average costs

The different topologies display a similar pattern with respect to distrust values. As shown in Figure 6.8 and the associated Table of average values, random networks are the most expensive with respect to distrust, and have the lowest number of distrusted links; linear networks remain constrained in the number of distrusted links, with costs decreasing; scale-free networks do not show a sensibly better behaviour, with comparable number of distrusted links and costs; and finally, total networks perform best, with the highest levels of links and relatively lower costs. As shown in the graph, the diverging behaviour of total and random networks is remarkable: the former have almost stable distrust costs with increasing distrusted links, while the latter have stable links with increasing costs.



	Average Distrust Links	Average Distrust Cost
Random	35.4	167.38
Linear	71.82	130.18
Scale-free	75.29	134.5
Total	264.13	103.1

Figure 6.8: Distrust distribution and average costs

The comparison between tables shows that the average number of trusted and distrusted links grows in parallel, while the related costs decrease in a similar vein across the different topologies. Nonetheless, this proportion is not linear. Trust propagates at a much higher rate than distrust in these balanced networks, and there is a small difference between scale-free and linear networks, where the former presents more distrust than trust cost when compared to the latter. These observations suggest that trust in scale-free networks is a more frequent and more relevant property in information transmission than in linear graphs in general, and that the latter are less affected than scale-free ones by distrust propagation.

We now briefly compare these experimental results with the meta-theoretical properties of the (`un`)`SecureND`$^{\texttt{sim}}$ structural derivations from Subsection 4.3.2. Lemma 2 in [Primiero *et al.* 2017] states that, given a fixed number of sceptic agents in a derivation, the resulting value of trust instances, defined as epistemic costs, is only due to the applications of the `verify_high` rule. The applications of the rule in question map directly to the number of order relations satisfied by agents in the derivation, and hence to the number of agents that are higher in the order than the agent appearing in the conclusion. Our experimental results show that this cost value is higher in random networks than in graphs with a linear order, where the latter correspond to derivations such that $\forall v_i, v_j \in V.(v_i < v_j) \vee (v_i > v_j)$. In the latter ones, a higher number of transitively valid relation (due to the totality of the graph) means fewer instances of the `verify_high` rule are applied. For the case of distrust costs, [Primiero *et al.* 2017, Lemma 3] states that, given a fixed number of lazy agents, the value of distrust instances is only due to the applications of the `unverified_contra` rule. The explanation above, mapping order relation to topologies, holds in this latter case as well.

6.2.3 Rankings

To analyse results based on the ranking of the seeding nodes, we consider the correlation between ranking and consensus in scale-free networks and investigate:

- ▷ whether a strictly higher ranking for one of the seeding nodes implies a greater chance to obtain a unanimous graph labelled by the same formula;
- ▷ which type of scale-free networks (among overly sceptic, balanced and overly lazy) has the higher probability to reach a unanimous graph from a seed with higher ranking.

The results of our analysis are plotted at the top of Figure 6.9. This plot reports the proportion of runs that reach consensus about a label from a higher ranked seed (the RHS axis). Results can be summarised as follows:

1. there is no strict correlation between a highly ranked seed and the labelling of the network: the number of cases where the consensus is reached *and* the label is the same as the one from the higher ranked seed is relatively small (min $\frac{1}{7}$, max $\frac{8}{26}$);

2. an overly sceptic scale-free network offers the highest probability to obtain a unanimous graph labelled with the input of the higher ranked node among the seeds; the comparison between the lazy and the balanced network sees the former obtain better results in general, and the latter only for significantly large networks.

The next evaluation of the data concerns the correlation of ranking of the seeds with costs: namely, we investigate whether information transmission from equally ranked seeds is more or less expensive than transmission from differently ranked ones. The results for overly lazy, balanced and overly sceptic scale-free networks are plotted at the bottom of Figure 6.9. The results can be summarised as follows:

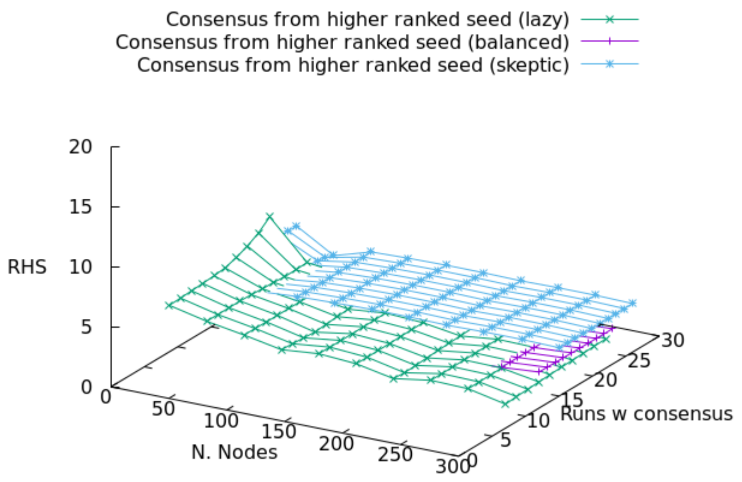
1. contradictory information transmission from differently ranked nodes tends to be more expensive than from equally ranked nodes in balanced and overly lazy networks: here the costs are induced by a less stable labelling for the information transmitted by higher ranked nodes;

2. in lazy networks, the higher costs of differently ranked seeds tend to collapse for maximally large networks, where the costs are less than the corresponding seeding with equally ranked nodes.

3. contradictory information transmission from equally ranked nodes tends to be more expensive than from differently ranked nodes in overly sceptic networks: this can be a symptom of the greater overall epistemic balance of the sources spreading information, combined with the more common attitude of agents to require confirmation.

6.2.4 Distrust and Epistemic Attitude

In the ensuing experiments, we focus on scale-free networks only and their distrust behaviour. First, we consider distrust as a parameter of the proportion of lazy agents in a network of 300 nodes, with a random assignment of labels to seeding agents (lazy/sceptic). As shown in Figure 6.10, there is a strict correlation between the proportion of sceptic and the distrust behaviour: the more lazy agents are present in the network, the higher its overall distrust value. While this is obvious in view of the algorithm design, it is interesting to remark that in the case of a fully sceptic network (where no lazy agents are allowed), the value of distrust is to be associated entirely with the presence of contradictory information, and hence it can be used as a parameter of contradiction



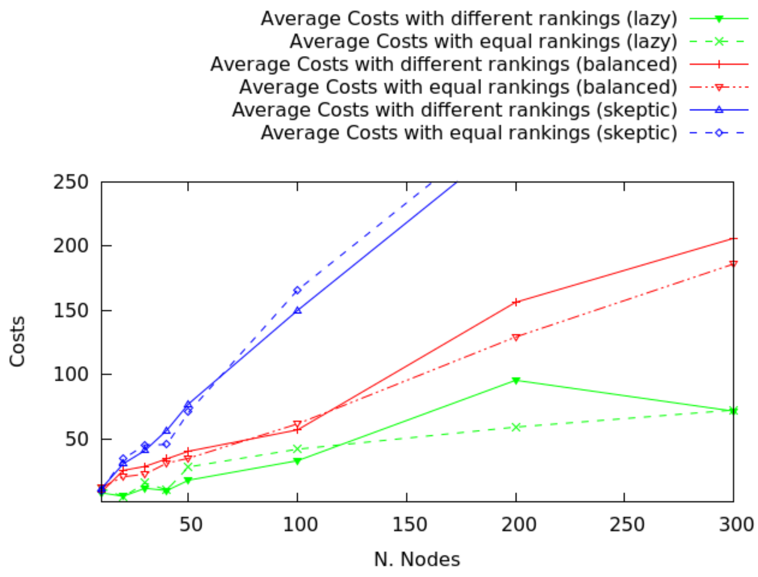
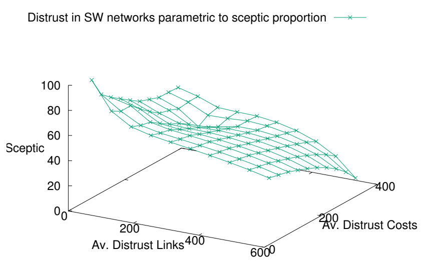


Figure 6.9: Top: Higher ranked seeds in consensus reaching network. Bottom: Costs and Ranking.

diffusion. The associated Table offers average values over 100 runs. It illustrates that conflict resolution is responsible on average for roughly 10% of the network's distrusted edges, with costs averaging at around a fraction $\frac{1}{7}$ of those of a highly lazy network (i.e., with 10% of sceptic agents).



% Sceptics	Av. Distrusted Links	Av. Distrust Cost
10%	575.45	350.92
30%	170.11	276.79
50%	129.98	213.36
80%	69.2	113.83
100%	32.25	45.35

Figure 6.10: Distrust behaviour and epistemic attitude

We now extract the values for a balanced network (i.e., with 50% of sceptic agents) and compare them to the initial distribution of seeds qualified as lazy-sceptic agents. As Figure 6.11 shows, there is a strict correlation of the final distribution of distrust values with the initial condition of the network: the range of minimal values for both distrust costs and number of distrusted links is relatively stable, while their maximum value decreases when moving from a configuration that has two sceptic agents as initial nodes to one that has two lazy ones. The result on distrust across the network is less influenced by the role of agents *distributing* the information than by the role of agents *receiving* it.

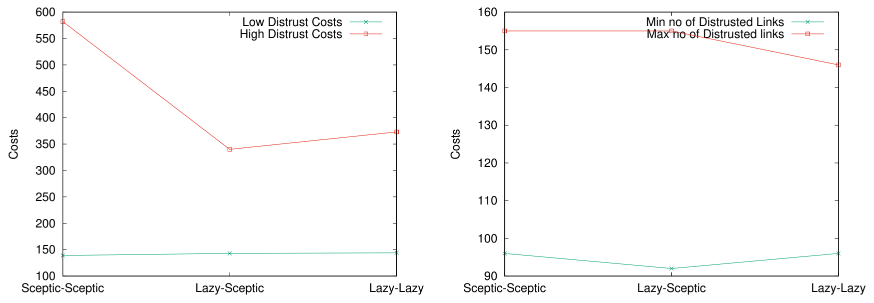
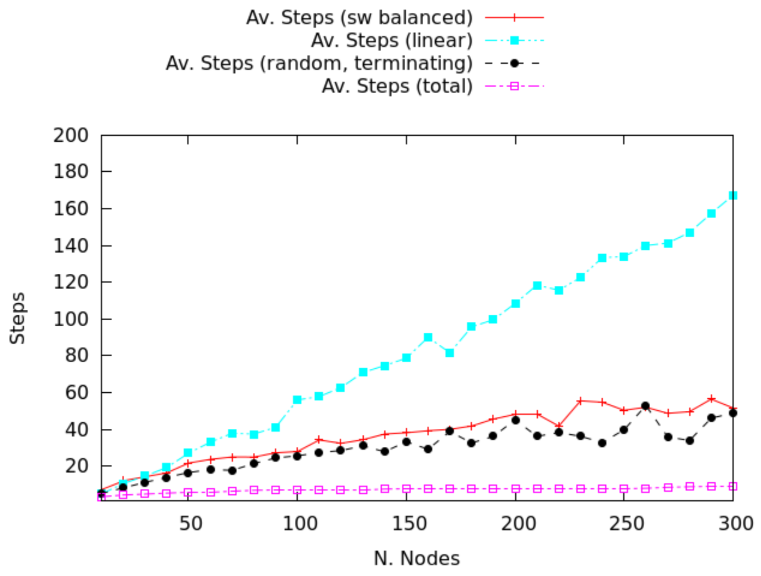


Figure 6.11: Initial nodes' epistemic attitudes and distrust

6.2.5 Time Complexity

The final analysis concerns the time complexity calculated as running time efficiency. Recall that each step in the simulation expresses one message transmission or equivalently a epistemic operation of verification or distrust. For the purposes of our experiments, running time complexity is computed as a function relating the size of the network (in terms of the number of nodes) with the number of time steps required for termination (i.e. to fully label the graph), using the same data for networks of fixed



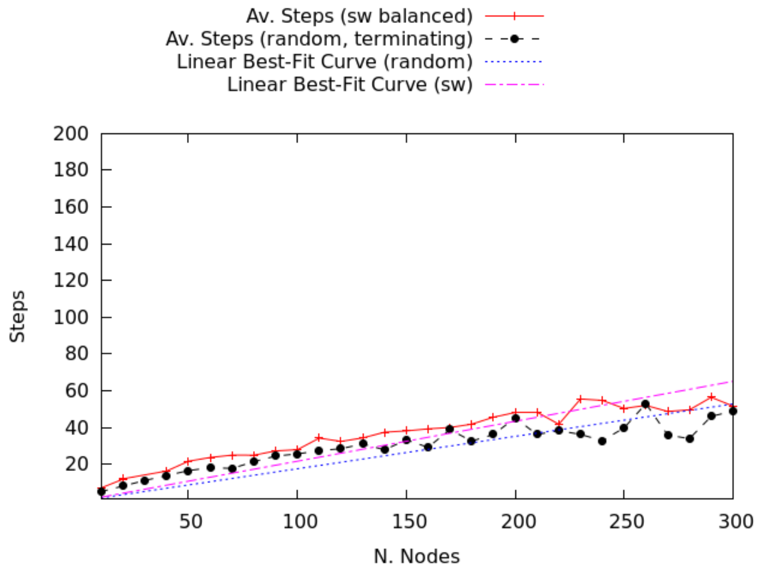


Figure 6.12: Top: Time complexity. Bottom: Best fit comparisons.

dimensions between 10 and 300 nodes as in Section 6.2, with a total of 30 data points spaced every 10 nodes. We wish to know whether and in which way the network topology affects this relation. We find that the computational complexity of the algorithm belongs to the linear class. In Figure 6.12, we concentrate on the average values of a balanced scale-free network and compare them against linear, random and total networks. This comparison is shown in the top graph of Figure 6.12. For the random network, in view of the time-out conditions mentioned in Subsection 6.2.1, the graph only reports the values for the terminating runs. The results can be summarised as follows:

1. Linear networks are the most computationally expensive in terms of the time it takes for the procedure to terminate;
2. Scale-free are at most as expensive as random ones;
3. Total networks have a linear increase of the computational complexity in the number of nodes and require the shortest time to terminate.

The difference between total networks (the cheapest ones) and linear networks (the most expensive ones) is over 150 steps. The algorithm has complexity $\mathcal{O}(n)$, see the dotted line of the bottom graph of Figure 6.12 for the best fit of a linear function to the data for scale-free and random networks.

Obviously, compared to the balanced network cases, time complexity of an overly lazy scale-free network will be lower than that of a balanced one, as a result of the lower number of confirmation steps that are required to fully label the graph. Correspondingly, time complexity of an overly sceptic network will be higher, as a result of the higher number of confirmation steps required to fully label the graph.

6.3 The Tangle DAG Multi-Agent Simulation

In this Section, we present and evaluate experimental results from the paper [Bottone *et al.* 2018] for the Tangle DAG random graph model simulations introduced in Section 5.4.

6.3.1 Initial Findings and Comparisons

The interface setup is shown in Figure 6.13, which also shows the Netlogo 3D View in action while the Tangle is being simulated using a spring-loaded layout started at the origin, and different colours for cutsets, and two graph and node statistics, namely the number of tips *not* and the cumulative weight *cw*. In our exploratory simulations, we replicated the theoretical intuitions of the whitepaper [Popov 2017] and the simulations of [Kusmierz 2017] in that the incoming tip process bounds the random fuctuations of the number of tips in $\mathcal{T}(t)$ in a narrow band around a multiplicative constant of λ and $\lambda\, t$ for the uniform random (Figure 6.14c) and MCMC tip selection strategies (Figure 6.14d), respectively; and that the cumulative weight of nodes grows linearly in time after the initial burn-in, as in Figure 6.14b, while the edge distribution shape (Figure 6.14a) remains mostly unchanged around an average of 4 as the Tangle grows. We also found that the Tangle samples have a pleasing visual structure, already apparent in Figure 5.11 of Section 5.4. The viewing perspectives of the NetLogo simulator also

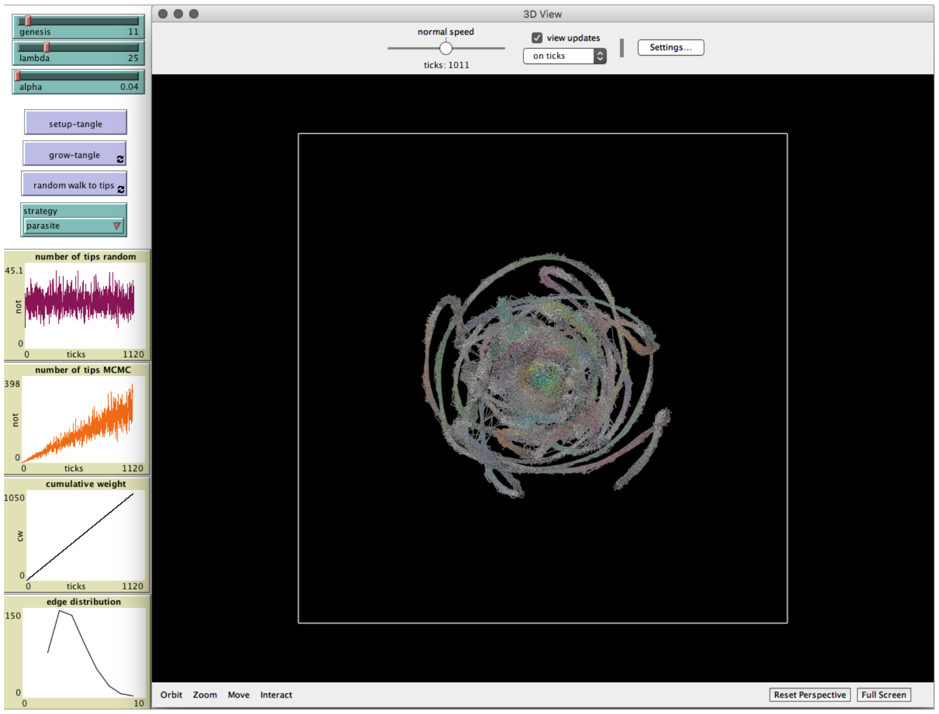


Figure 6.13: The NetLogo Interface View with a 3D rendering

simplify inspection of nodes when looking for further structure, enabling zooming in and out of the current visualisation, and the `nw` and `filename` extensions provide facilities for saving data in `graphml` and text file format for further processing or for using the simulations as graphical environments for more complex agent strategies.

6.3.2 Conjectures

The preliminary analysis of the Tangle $\mathcal{T}$ in NetLogo has also revealed an interesting phenomenon, which is more prominent for some values of α in the range $0.01 - 0.7$. The DAG structure naturally induces a hierarchy of cutsets in terms of the variable cw, which reveals how the parameter λ really drives the width of the network, acting as a bottleneck when the random clock creates a low number of new nodes. Figure 6.15 gives a clearer view of this phenomenon.

In addition to what is currently known about the desirability of exponentially biased tip selection strategies as studied in [Popov 2017, Popov *et al.* 2017], we would like to find out if there is some optimality principle to drive tip selection. We first observe that the architecture of the Tangle Markov Chain can be viewed as a special case of a directed network that has become commonplace in the Deep Learning literature, where each hidden layer is one state of the chain, and the initial genesis transaction is in a par-

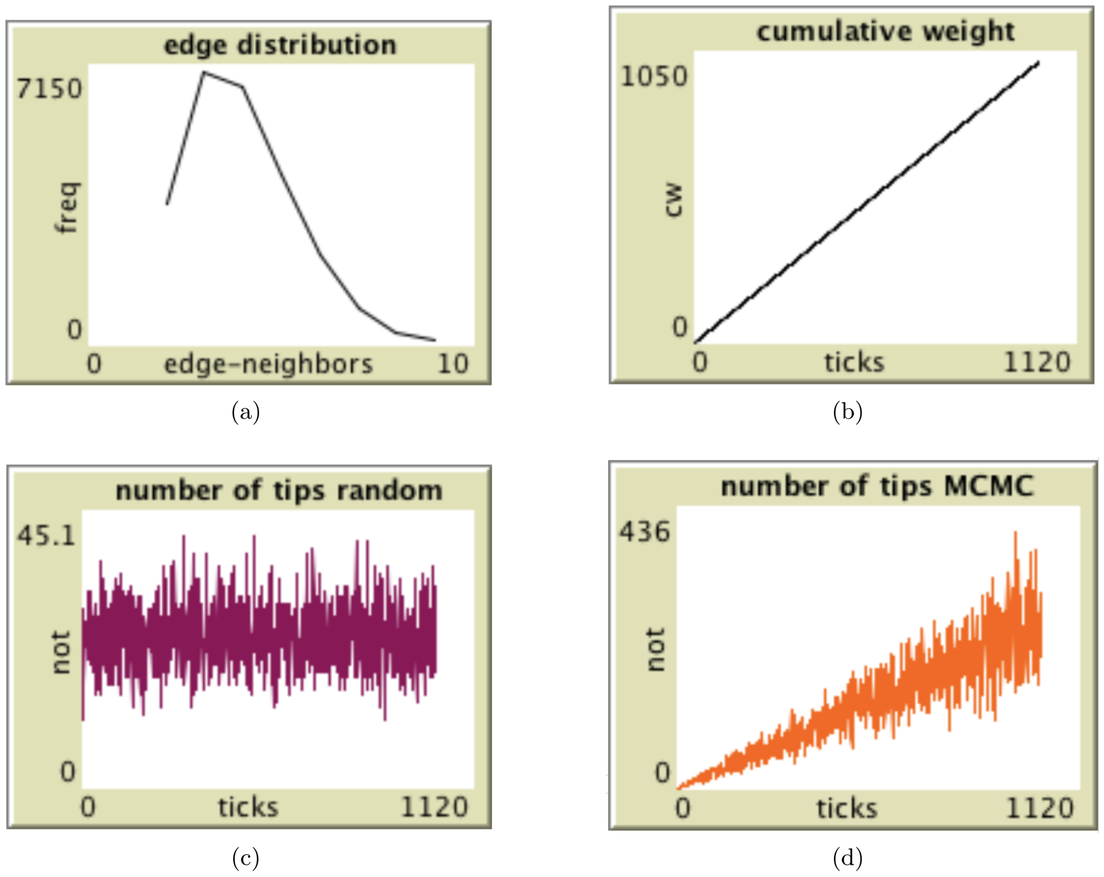


Figure 6.14: Statistics from a random sample of 25256 nodes, stopped at $\mathcal{T}_{1010}$, $\rho = 10$, $\lambda = 25$, $\alpha = 0.01$. (a) Edge distribution: $d(v) := 2 + d_{\mathbf{in}}(v)$. (b) *cw* of genesis node at $\mathcal{T}_{1010}$, $\rho = 10$, $\lambda = 25$, $\alpha = 0.01$. The other nodes have the same behaviour. (c) Number of tips after $\mathcal{T}_{1010}$ $\rho = 10$, $\lambda = 25$, $\alpha = 0.01$ stays on average around λ. (d) Number of tips after $\mathcal{T}_{1010}$, 25256 nodes. $\rho = 10$, $\lambda = 25$, $\alpha = 2$ drifts linearly in the number of ticks.

ticular sense the higher-level representation of the information stored in the distributed ledger $\mathcal{T}$. Such networks are often randomly initialised and updated. The Tangle of course is not a collection of neurons, but it is very much an artificial information-processing structure. The transition probability in (4.1) of Subsection 4.4.4 provides a mean-field criterion for signal propagation deep into the network, as recently discovered for DNNs by [Shoenholz *et al.* 2017]. An information-theoretical explanation called the *information bottleneck* based on the Bellman optimality principle has been suggested by [Schwartz-Ziv & Tishby 2017]. If we denote with T a compressed variable, such an algorithm minimises $\min_{p(t|x)} I(X;T) - \beta I(T;Y)$ with $I(X;T)$ and $I(T;Y)$ the *mutual information* between $X;T$ and $T;Y$, respectively, and β a Lagrange multiplier, which

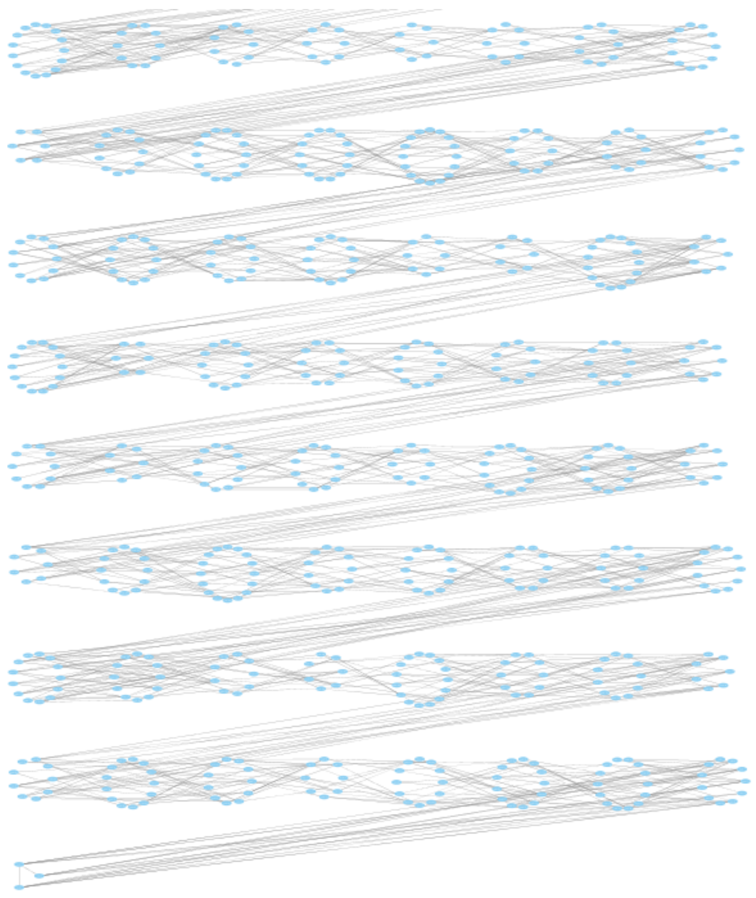


Figure 6.15: Bottleneck views in the *igraph* visualisation package

provides a growth bound for action by independent agents and walkers on the structure [Tishby & Polani 2010]. Just as a DNN is designed to learn how to describe a feature X to predict a function Y and eventually compress X to only hold the information related to Y, in the same vein one might arrive at an efficient, robust representation of information in the distributed ledger that is expressive, relevant information-rich and resistant to attack and propagates information in the network efficiently and instantaneously [Poole *et al.* 2016].

6.3.3 Multi-agent Analysis of Parasite Strategies

Currently, our simulated multi-agent system only implements a mixture of naive parasite strategies such as building an offline linear network and incrementally attaching it to the main Tangle at successive, evenly-spaced points, which tend to have a negligible effect for sufficiently large λ. A further step for the analysis presented here, which we reserve for future work, is to study the evolution of the Tangle information structure under sophisticated attack vectors, which is often found in biological virus attacks. For example, in Section 4.2 of [Popov 2017], a possible splitting attack scheme towards a Tangle network implementing a MCMC algorithm based on dynamic load-balancing of different branches containing conflicting information is mentioned; these complex strategies could be modelled in our framework by using NetLogo to simulate the environment and implementing agents as malicious nodes using a high-level planning framework based on the Belief-Desire-Intention architecture, either in NetLogo directly or by linking with the Jason development environment as recently done in [Ramirez & Fasli 2017]. Additionally, we consider endowing the nodes of $\mathcal{T}$ with further internal variables, such as smart contract items γ that can be used to deal with trust and distrust, and study contradictory information resolution alongside the approach adopted in [Primiero *et al.* 2017].

Chapter 7

Conclusion and Further Work

In this thesis, we have presented our progress towards a practical implementation of scalable and extensible information-based multi-agent systems within the framework of *agoric computation*. In what follows, we take stock of what has been achieved with respect to the central aim of the body of work underlying this thesis, and offer concluding remarks and suggestions for future work.

7.1 The State of Agoric Computation

In this Section, we give a brief summary of the main contributions of our thesis with respect to the research aims and how they have been supported.

7.1.1 Publish-Subscribe Patterns

In particular, we have established that the publish-subscribe programming pattern, when coupled with open-source multi-agent development environments and rule-based programming, provides a viable infrastructure for building abstractions based on the context and theories reviewed in Chapters 2 and 3, and that once one addresses information flows in open systems within a coherent framework for filtering raw facts and local data into higher-order representations describing the overall functioning of the system, one can build robust, effective systems that mimic real world behaviour. Clearly, the way components and agents 'talk' to each other in the pursuit of collective computation via lightweight messaging protocols is an essential part of any engineered system consisting of a large number of components, providing a common language for computation in much the same way, for instance, electrical impulses via the central nervous system provide a communication language for coordination and effecting and feeding back local changes in the human body that dispenses with the low-level subsystems occurring in the central and peripheral parts.

7.1.2 Trust in Cyber-Physical Systems

We have also shown that simple mathematical rules for consistency based, for instance, on the natural deduction calculus allow for contradiction handling and assessing the epistemic cost of trust under computational constraints the context of typical, fixed network models that can be easily simulated and used to monitor and analyse the emergence of trust as an overall assessment of agoric systems and the effect of contradictory information in the presence of various proportions of epistemic attitudes possessed by agents. In addition, we have shown that starting from simple selection rules for one can build large yet relatively lightweight dynamic graphs (or *ledgers*) for embodying trust and cross-verifying a growing body of transactions, which holds promise for building trusted data interchange. More work, of course, needs to be done to confirm the validity of these computational model for trust but, at its face value, the basic idea is appealing: that by pruning, selection and variation/exploration it is possible to build robust systems that align incentives, extract actionable information and replicate some of the attractive real-world characteristics of marketplaces in the face of a deluge of data, bringing to fruition the thus far unrealised vision of agoric computing.

7.1.3 Facets of Information

More generally, the variety of forms that 'information' may take in an agoric system is one of the strengths of this approach. Out of an infinite choice of possible modalities, we have specifically addressed three special cases: those of information carried, received and transmitted by agents as a numerical quantity, the case of information as a logical statement and its negation (coupled with binary epistemic modalities for agents), and that of information as an internal variable for accumulated, cross-verified transaction graphs in a trust network. Work is also underway to apply the principles presented in this thesis to a computationally grounded formalism to model trust as currency in a system of agents, mentioned in Section 7.3.

7.2 Limitations of the Current Approach

No discussion would be complete without an overview of the overall limitations of the agoric computation framework. Three critiques could be advanced as to the present direction.

7.2.1 Simplicity

The first one is the valid criticism that by focusing on simple, rule-based approaches, the resulting engineered systems end up being too simple to be relevant, since in real-life systems and production software feedback loops, bugs and errors reign supreme, and – for instance – sensor data and transaction handshakes exhibit a lot of noise. In the case of the natural deduction calculus-based approach, for instance, a spectrum of epistemic modalities and rankings of agents has yet to be attempted, and it is not as clear-cut that the same results would hold for the kind of networks we study. We counter that the vast majority of complex systems often descend from simple rules and advance by a thousand cuts by Ockham's razors; in the field of software engineering, for example, it is hard to beat in terms of reliability the code[1] that has sent in space both Saturns and the Space Shuttles in the last four decades, which is very simple, almost error-free code and has been recently published on GitHub for the Apollo 11[2]. Many other natural and artificial systems achieve great complexity despite the limited set of actions available

[1]In contrast to software, hardware fails surprisingly often in mission-critical contexts.
[2]https://github.com/chrislgarry/Apollo-11

to their subcomponents, and one could contend that decomposability of rules and high level function is almost a necessary condition for expressive, robust systems. Relatedly, the case of agoric systems where the number of components varies substantially in any direction as time progresses is more difficult to study and formalise than systems based on a fixed or growing number of components; an example for data is given by RAID architectures, where the main aim is to preserve the integrity and contents of the system rather than achieving internal representation; such scenarios would constitute a rigorous testbed of the agoric approach.

7.2.2 Languages

The second, related critique concerns the software tools that have been used to validate this approach. In principle, any Turing-complete programming language can be used to formally validate the agoric computation approach, and one could contend that a large part of the performance of a system reflects the implementation of the software tool used rather than the multi-agent system *per se*. Despite the common, Java virtual machine-based basis of the systems we studied, one could easily counterpoint that using two software simulation environments to test what should be a unifying framework is a bit of an odd choice. However, our framework for agoric computation is not defined by a simple language for expressing rules, but instead expresses a *methodology* and a set of principles for constructing robust, relevant abstractions that are nevertheless grounded in real facts and data.

7.2.3 Verification

The third critique which we envisage is that we have said little in the way of mathematical verification and model-checking in the body of work presented in this thesis. However, its main focus lies elsewhere and a self-contained discussion of these techniques would easily take up a dedicated monograph. We point out that the same variety of forms for information that we have addressed in the archetypal models of Chapters 4 and 5 is easily amenable to the automated validation techniques inherent in model checking and automated theorem proving because of how agoric processes are constructed. At its core, verification takes a set of inputs such that a logical statement, program or theorem, parses it in a prover or checker and produces a set of outputs – which could be a set of binary outcomes such as valid/not valid, or a formal proof, or a counterexample. All rules, statements and programs must be stated in formal logic, where the assumptions are apparent. This thesis has briefly touched on the practical applicability of proof systems such as propositional, natural deduction, higher order logic and BDI logics, which can fruitfully be subject to techniques such as automated resolution [Harrison 2009], process calculi, and Gentzen calculi [Poggiolesi 2011], and these can be used to prove the robustness of the agoric approach. Future work should, in due course, look at the formal specification of agoric multi-agent models in such a way that they behave as expected at large scale – in the ballpark of millions rather than the hundreds and thousands studied here – while still maintaining their nimbleness.

7.3 Future Directions

In the preceding Section 7.1, we have mentioned work on a computationally grounded system for trust that is currently in preparation and review. In the context of banking networks and autonomous interactive systems, trust-derived concepts and informational efficiency are crucial. We describe this in more detail in the following notes, along with related thematic areas that we are currently investigating.

In particular, the following work is being pursued:

▷ *A computationally grounded formalism to model trust in a system of agents.* Here trust is expressed by a function derived from the local states of agents and from the observable states of other agents within the semantics of Interpreted Systems [Fagin *et al.* 1995]. This approach allows the simulation of trust relations, and trust thresholds for related fiduciary concepts such as untrust, mistrust, distrust between agents and a detailed analysis of their dynamics, where the focus is on the interplay between the flow of information in a collection of autonomous yet interacting entities, and some properties of the system such as opacity[3] so that the sudden collapse of trust in networks can be studied; this is useful, for instance, for simulating regulatory scenarios for government officials. Money, or its time equivalent, is as good a yardstick to measure trust [Simmel 1907], and it has been one of the insights of [Minsky 1986] that assigning mental currencies to loosely comparable alternatives competing for certain limited resources in a society of agents is one explicit way to compute things and study their evolution. Our starting point is [Yasutomi 1995]'s classic conceptual model of the emergence of money, whereby agents $\{Ag_1, \ldots, Ag_N\}$ produce one type of item (or good, or token) to be consumed, which might be an item of information of something else that engenders trust (for simplicity, we suppose that agent i produces item i). Agents interact via random search of the agent space and elementary pairwise transactions where they exchange items according to specific rules and take turns, and the state of the system evolves via three bookkeeping vectors of holdings, demand, and a world-view of how trusted another agent is, which models their preferences; such a system has been replicated for different network topologies by [Paolucci 2013], and we build on this by stressing the temporal, informational element to fiduciary modelling. Trust is distinct from reputation in that the latter is a belief about

[3]Or transparency: i.e. the degree by which something is visible to other components.

some known past attribute[4]; in contrast, trust is forward-looking, and encodes a belief that some trusted agent will not exploit the truster's dependence and vulnerability[5], and can be conceptually imagined as the mental process of leaping, via a suspension of belief, across the chasm of the unknowable from an interpretation (and experience) to a state of expectation (positive in the case of trust, negative for distrust) towards other agents' actions and intentions [Nooteboom 2002]. This extrapolation, or "leap of faith" makes it both more difficult to measure, and more interesting to study from a computational, information theoretic perspective since trust and trustworthiness are intimately connected with how to model new situations and possibly changes in the environment; this is reflected by extending the world view of agents by a further bookkeeping vector encoding their future expectation of trustworthiness.

- *Lattice based models and propagator networks.* In previous unpublished work, we explored the feasibility of the lattice theory of information based on discrete, decomposable combinatorial structures [Shannon 1953], which is sufficiently general to serve as a basis for models of phenomena in an arbitrarily structured and countable space of possibilities[6]. In this formulation, information is *something that has a* **value** *and is* **common** *to agents* $a \in \mathbf{Ag}$ *at a given time*, i.e. an equivalence class of all ways of describing the same information possessed by agents. Such equivalence classes serve as the basic information elements of a system, and are parametrised by set partitions whose algebraic lattice superstructure satisfies a closure and consistency property. Values can be positive, or negative, and roughly correspond to perceptions of benefit and cost. Values embody variety and diversity within the environment; the key assumption is that a quantitative, universal, and domain independent representation of value is required for a model of exchange, just like in Minsky's society, and serves as a store of information that can be used for integration and transformation. Beliefs about these values can be quantified within a categorical, logical, numerical or symbolic system. Beliefs about values, when mapped to a probability space, give rise to distributions. and one can build a quite powerful mathematical theory of integrated dynamical systems using, for instance, partition-dependent functional stochastic integration [Rota & Wallstrom 1997], which turns out to be quite useful in computation and concurrency. More recently, lattice-based data structures have been used by [Kuper & Newton 2013] in the context of deterministic parallel programming,

[4]Which can be analysed in hindsight, weighing capabilities and statistics, and common pitfalls such as adverse selection mitigated.

[5]And thus can be analysed as expectation, and its pitfalls as moral hazard long studied using game-theoretical techniques.

[6]As such it was conceived by Shannon as a theory of information for communication problems in which there exist a possibly large number of information sources simultaneously in operation.

and several other authors, e.g. [Kmett 2017, Meiklejohn & Van Rooy 2015] have noticed that the propagators models of Subsection 2.1.7 in Chapter 2 represent monotone function between join-semilattices, thus one can recast a propagator network as a hypergraph with propagators for hyperedges, and join-semilattices for nodes, thus unifying several of the mathematical formulations of Chapter 2. One useful property is that if every semilattice is finite, naive propagation terminates and yields a deterministic, rather than nondeterministic result, regardless of scheduling strategy, redundant firings, or evaluation order: conflict-free Convergent Replicated Data Types (CRDTs), Datalog, SAT solving, functional reactive programming, and constraint programming all fit into this mould.

▷ *The PAC approach to information flow in a dynamic network.* In this vein, we mentioned the information bottleneck theory in the context of the evaluation of the DAG structure of our work on distributed ledgers in Subsection 6.3.2. The *perception-action loop* [Tishby & Polani 2011] provides a principled information-theoretical framework for sensing and acting under information constraints, and thus a path to easily simulate biological and AI-inspired systems, by replicating the feedback loop of reasoning organisms in real life: that is, most goal-oriented systems in noisy, constantly changing environments balance the cost of storing information about the past against the payoff of achieving the desired goals in the future, using standard mathematical concepts such as mutual information, distortion-rate theory, the Bellman-Laplace equation for first-order conditions and Partially Observable Markov Decision Processes (POMDP). In particular, one could generalise POMDP[7] to allow the agents to update their policies based on new data as the simulation (or time) unfolds, of otherwise handle unexpected contingencies as conditions change, which fits broadly with the areas of study of this thesis. as in the wider literature in cognitive agents, the focus is on perceptions-goal oriented hypotheses generation, directed by active predictions and useful decisions that are then tested by external o internal information gathering. The main aim is to reduce decision complexity, maximise the environment information gain, and increase robustness to model fluctuations.

[7]See also, from a model checking perspective, [Kwiatkowska 2013].

Bibliography

[Abbass *et al.* 2016] Hussein A Abbass, George Leu and Kathrin Merrick. *A Review of Theoretical and Practical Challenges of Trusted Autonomy in Big Data.* IEEE Access, vol. 4, pages 2809–2830, 2016. (Cited on page 25.)

[Abbass *et al.* 2018] Hussein A Abbass, Jason Scholz and Darryn J Reid, editors. Foundations of trusted autonomy, volume 117 of *Studies in Systems, Decision and Control*. Springer, 2018. (Cited on page 25.)

[Abelson *et al.* 1996] Harold Abelson, Gerald Jay Sussman and Julie Sussman. The structure and interpretation of computer programs. MIT Press, 2nd édition, 1996. (Cited on page 23.)

[Abramsky 1997] Samson et al. Abramsky. *Semantics of Interaction: An Introduction to Game Semantics.* In Semantics of Logic and Computation. Publications of the Newton Institute, 1997. (Cited on page 23.)

[Akkaya 2016] Ilge Akkaya. *Data-Driven Cyber-Physical Systems via Real-Time Stream Analytics and Machine Learning.* PhD thesis, EECS Department, University of California, Berkeley, http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-159.html, October 2016. (Cited on page 7.)

[Aldous & Lanoue 2012] David Aldous and Daniel Lanoue. *A lecture on the averaging process.* Probability Surveys, vol. 9, pages 90–102, 2012. (Cited on page 19.)

[Aldous *et al.* 2014] David Aldous, Daniel Lanoue and Justin Salez. *The Compulsive Gambler Process.* arXiv:1406.1214 [math.PR], 2014. (Cited on page 19.)

[Aldous 2013] David Aldous. *Interacting Particle Systems as Stochastic Social Dynamics.* Bernoulli, vol. 19, no. 4, pages 1122–1149, 2013. (Cited on page 19.)

[Androutsopoulos *et al.* 2018] K Androutsopoulos, L Aristodemou, J Boender, M Bottone and E Al. *MIRTO: an Open-Source Robotic Platform for Education.* In European Conference of Software Engineering Education (ECSEE2018), 14-15 June 2018 (Accepted/In press), June 2018. (Cited on page 83.)

[Arthur 2016] T W Richard Arthur. Natural deduction: An introduction to logic with real arguments, a little history, and some humour. Broadview Press, 2016. (Cited on page 35.)

[Attie & Lynch 2016] Paul C Attie and Nancy A Lynch. *Dynamic input/output automata: A formal and compositional model for dynamic systems.* Information and Computation, vol. 249, pages 28–75, August 2016. (Cited on page 21.)

[Awodey 2006] Steve Awodey. Category theory. Oxford University Press, 2006. (Cited on page 22.)

[Baier & Katoen 2008] Christel Baier and Joost-Pieter Katoen. Principles of model checking. MIT Press, 2008. (Cited on page 8.)

[Balduzzi 2014] David Balduzzi. *Cortical Prediction Markets.* In Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Paris, France, May 5-9, pages 1265–1272, 2014. (Cited on page 23.)

[Balduzzi 2016] David Balduzzi. *Grammar for Games: a Gradient-Based, Game-Theoretic Framework for Optimization in Deep Learning.* Frontiers in Robotics and AI, vol. 2:39, pages 1–14, January 2016. (Cited on page 23.)

[Barabasi 2015] Albert Barabasi. Network science. Cambridge University Press, 2015. (Cited on pages 39 and 46.)

[Barbieri & Mascardi 2011] Matteo Barbieri and Viviana Mascardi. *Hive-BDI: Extending Jason with Shared Beliefs and Stigmergy.* In ICAART (2), pages 479–482. Citeseer, 2011. (Cited on pages 66 and 67.)

[Barbon *et al.* 2016] Gianluca Barbon, Michael Margolis, Filippo Palumbo, Franco Raimondi and Nick Weldin. *Taking Arduino to the Internet of Things: the ASIP programming model.* Computer Communications, 2016. (Cited on page 65.)

[Baum 2003] Eric B Baum. What is thought? MIT Press, 2003. (Cited on pages 10 and 11.)

[Beni 2004] Gerardo Beni. *From swarm intelligence to swarm robotics.* In International Workshop on Swarm Robotics, pages 1–9. Springer, 2004. (Cited on page 64.)

[Besnard & Hunter 2008] Philippe Besnard and Anthony Hunter. Elements of argumentation. MIT Press, 2008. (Cited on page 26.)

[Birman & Joseph 1987] K. Birman and T. Joseph. *Exploiting Virtual Synchrony in Distributed Systems.* SIGOPS Oper. Syst. Rev., vol. **21**, no. 5, pages 123–138, November 1987. (Cited on pages 24 and 31.)

[Bollobás 1998] Bela Bollobás. Modern graph theory. Springer, 1998. (Cited on page 39.)

[Bonabeau 1999] Eric Bonabeau. *Editor's introduction: stigmergy.* Artificial Life, vol. 5, no. 2, pages 95–96, 1999. (Cited on page 64.)

[Bordini *et al.* 2007] R. H. Bordini, J. F. Hübner and M. J. Wooldridge. Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology). John Wiley & Sons, 2007. (Cited on pages 32 and 83.)

[Bordini *et al.* 2009] Rafael H Bordini, Mehdi Dastani, Jürgen Dix and Amal El Fallah Seghrouchni, editors. Multi-agent programming: Languages, tools and applications. Springer, 2009. (Cited on page 34.)

[Bottone *et al.* 2016a] Michele Bottone, Filippo Palumbo, Giuseppe Primiero, Franco Raimondi and Richard Stocker. *Implementing Virtual Pheromones in BDI Robots Using MQTT and Jason (Short Paper).* In 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), 03-05 Oct 2016, Pisa, Italy., pages 196–199, 2016. (Cited on pages 62, 64, 82, 83 and 84.)

[Bottone *et al.* 2016b] Michele Bottone, Giuseppe Primiero, Franco Raimondi and Vincenzo De Florio. *A Model For Trustworthy Orchestration in the Internet of Things.* In Proceedings of the 12th International Conference on Intelligent Environment 2016, London (UK), 14-16 Sept. 2016, pages 171–174, 2016. (Cited on pages 44, 45, 62, 68, 82, 84 and 85.)

[Bottone *et al.* 2016c] Michele Bottone, Giuseppe Primiero, Franco Raimondi and Neha S Rungta. *From Raw Data to Agent Perceptions for Simulation, Verification, and Monitoring.* In 12th International Conference on Intelligent Environment 2016:- 5th International Workshop on Reliability of Intelligent Environments (WoRIE'16), pages 66–75. IOS Press, 2016. (Cited on pages 42, 62 and 82.)

[Bottone *et al.* 2018] Michele Bottone, Franco Raimondi and Giuseppe Primiero. *Multi-agent based simulations of block-free distributed ledgers.* In Accepted In: E3WSN, Co-located with the 32nd IEEE Conference AINA-2018, 16-18 May 2018, Pedagogical University of Cracow, Poland, 2018. (Cited on page 98.)

[Boyen *et al.* 2017] Xavier Boyen, Christopher Carr and Thomas Haines. *Blockchain-Free Cryptocurrencies: A Framework for Truly Decentralised Fast Transactions.* https://eprint.iacr.org/2016/871.pdf, 2017. (Cited on page 54.)

[Bratmann 1999] M.E. Bratmann. Intention, Plans, and Practical reason. Cambridge University Press, 1999. (Cited on pages 6 and 32.)

[Bühlmann 1979] Niklas Bühlmann. Trust and power. Wiley, 1979. (Cited on page 26.)

[Burnett *et al.* 2011] Chris Burnett, Timothy J Norman and Katia Sycara. *Trust Decision-Making in Multi-Agent Systems.* In IJCAI2011 – 22nd International Joint Conference on Artificial Intelligence, 2011. (Cited on page 26.)

[Buzsaki 2006] Gyorgy Buzsaki. Rhythms of the brain. Oxford University Press, 2006. (Cited on page 11.)

[Cardelli & Gordon 1998] Luca Cardelli and Andrew D Gordon. *Mobile Ambients.* Electronic Notes in Theoretical Computer Science, vol. 10, pages 198–201, 1998. (Cited on page 19.)

[Cardelli & Gordon 2000] Luca Cardelli and Andrew D Gordon. *Mobile Ambients.* Theoretical Computer Science, vol. 240, no. 1, pages 177–213, June 2000. (Cited on page 19.)

[Castelfranchi & Falcone 2001] Cristiano Castelfranchi and Rino Falcone. Trust and deception in virtual societies, chapter Social Trust: A Cognitive Approach, pages 55–90. Springer, 2001. (Cited on page 27.)

[Castelfranchi & Falcone 2010] Cristiano Castelfranchi and Rino Falcone. Trust theory: A socio-cognitive and computational model. Wiley, 2010. (Cited on pages 26 and 27.)

[Chastain *et al.* 2013] Erick Chastain, Adi Livnat, Christos Papadimitriou and Umesh Vazirani. *Algorithms, games, and evolution.* Proceedings of the National Academy of Sciences, vol. 111, no. 29, pages 10620–10623, 2013. (Cited on page 9.)

[Chazelle 2012] Bernard Chazelle. *Natural Algorithms and Influence Systems.* Communications of the ACM, vol. 55, no. 12, pages 101–110, 2012. (Cited on page 21.)

[Chen *et al.* 2013] Taloue Chen, Vojtech Forejt, Marta Kwiatkowska, David Parker and Aistis Simaitis. *PRISM-games: A Model Checker for Stochastic Multi-Player Games.* In Proceedings of TACAS'13, 2013. (Cited on page 23.)

[Churyumov 2015] Anton Churyumov. *Byteball: A Decentralized System for Storage and Transfer of Value*, 2015. (Cited on page 54.)

[Clancey *et al.* 1998] W. J. Clancey, P. Sachs, M. Sierhuis and R. Van Hoof. *Brahms: Simulating Practice for Work Systems Design.* International Journal of Human-Computer Studies, vol. **49**, no. 6, pages 831–865, 1998. (Cited on page 33.)

[Da Costa Pereira *et al.* 2015] Célia Da Costa Pereira, Andrea Tettamanzi and Serena Villata. *A Computational Model of Trust Based on Message Content and Source.*

In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15), pages 1849–1850, 2015. (Cited on page 27.)

[Da Gama Batista *et al.* 2015] J Da Gama Batista, J-P Bouchaud and D Challet. *Sudden Trust Collapse in Networked Societies*. The European Physical Journal B, vol. 88, no. 10.1140/epjb/e2015-50645-1 55, 2015. (Cited on page 27.)

[De Florio & Primiero 2015] Vincenzo De Florio and Giuseppe Primiero. *A method for trustworthiness assessment based on fidelity in cyber and physical domains*. CoRR, vol. abs/1502.01899, 2015. (Cited on pages 44, 45, 70 and 71.)

[Derler *et al.* 2012] Patricia Derler, Edward A Lee and Alberto Sangiovanni Vincentelli. *Modeling Cyber–Physical Systems*. Proceedings of the IEEE, vol. 100, no. 1, pages 13–28, January 2012. (Cited on page 7.)

[DeVille & Lerman 2011] R.E. Lee DeVille and Eugene Lerman. *Dynamics on Networks I: Combinatorial Categories of Modular Continuous-Time Systems*. arXiv:1008.5359v2 [math.DS], 2011. (Cited on page 22.)

[DuBois *et al.* 2011] Thomas DuBois, Jennifer Golbeck and Aravind Srinivasan. *Predicting Trust and Distrust in Social Networks*. In Privacy, Security, Risk and Trust (PASSAT) and IEEE Third Inernational Conference on Social Computing (SocialCom), pages 418–424, 2011. (Cited on page 27.)

[Fagin *et al.* 1995] Ronald Fagin, Joseph Y Halpern, Yoram Moses and Moshe Y Vardi. Reasoning about knowledge. MIT Press, 1995. (Cited on page 107.)

[Flack 2017] Jessica C Flack. *Coarse-graining as a downward causation mechanism*. Philosophical Transactions of The Royal Society A, vol. 375, no. 20160338, 2017. (Cited on page 5.)

[Floridi 2010] Luciano Floridi. Information: A very short introduction. Oxford University Press, 2010. (Cited on page 10.)

[Foundation 2017] IOTA Foundation. *IOTA: A Cryptocurrency for the Internet of Things*, 2017. (Cited on pages 54 and 59.)

[Fournet & Gonthier 1996] Cédric Fournet and Georges Gonthier. *The reflexive chemical abstract machine and the join-calculus*. In Proceedings of 23rd ACM Symposium on Principles of Programming Languages (POPL'96), St Petersburg, FLA, pages 372–385, 1996. (Cited on page 20.)

[Fournet *et al.* 1996] Cédric Fournet, Georges Gonthier, Jean-Jacques Levy, Luc Maranget and Didier Rémy. *A Calculus of Mobile Agents*. In CONCUR'96

– Proceedings of the 7th International Conference on Concurrency Theory, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421. Springer-Verlag, 1996. (Cited on page 20.)

[Fudenberg & Tirole 1991] Drew Fudenberg and Jean Tirole. Game theory. MIT Press, 1991. (Cited on page 23.)

[Gambetta 1990] Diego Gambetta, editor. Trust: Making and breaking cooperative relations. Blackwell, 1990. (Cited on page 26.)

[Gambetta 2001] Diego Gambetta. *Can we trust trust?* In Diego Gambetta, editor, Trust: Making and Breaking Cooperative Relations, chapter 13, pages 213–237. Basil Blackwell, 2001. (Cited on page 26.)

[Golbeck 2006] Jennifer Golbeck. *Combining Provenance with Trust in Social Networks for Semantic Web Content Filtering.* In IPAW'06 – Proceedings of the 2006 international conference on Provenance and Annotation of Data, Chicago, IL –May 03 - 05, 2006, pages 101–108. Springer Berlin / Heidelberg, 2006. (Cited on page 26.)

[Goldin *et al.* 2006] Dina Goldin, Scott A Smolka and Peter Wegner, editors. Interactive computation: The new paradigm. Springer-Verlag Berlin Heidelberg, 2006. (Cited on pages 4 and 9.)

[Graeber 2011] David Graeber. Debt: The first 5000 years. Melville House Publishing, 2011. (Cited on page 53.)

[Grassé 1959] PIerre-P Grassé. *La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs.* Insectes sociaux, vol. 6, no. 1, pages 41–80, 1959. (Cited on page 64.)

[Guha *et al.* 2004] R Guha, Ravi Kumar, Prabhakar Raghavan and Andrew Tomkins. *Propagation of Trust and Distrust.* In WWW2004, May 17–22, New York NY, pages 403–412, 2004. (Cited on page 27.)

[Hang *et al.* 2009] Chung-Wei Hang, Yonghong Wang and Munindar P Singh. *Operators for Propagating Trust and their Evaluation in Social Networks.* In Decker, Sichman, Sierra and Castelfranchi, editors, Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAAMAS 2009), pages 1025–1032, 2009. (Cited on page 27.)

[Harrison 2009] John Harrison. Handbook of practical logic and automated reasoning. Cambridge University Press, 2009. (Cited on page 106.)

[Hayek 1937] F A Hayek. *Economics and Knowledge.* Economica, vol. IV (new series), pages 33–54, 1937. (Cited on page 9.)

[Hayek 1945] F A Hayek. *The use of knowledge in society.* American Economic Review, vol. 35, no. 4, pages 519–530, 1945. (Cited on page 9.)

[Hayek 1952] F A Hayek. The sensory order. University of Chicago Press, 1952. (Cited on page 9.)

[Hebb 1949] Donald O Hebb. The Organization of Behavior: a Neurophysiological Theory. John Wiley & Sons, 1949. (Cited on pages 9 and 11.)

[Herzig *et al.* 2008] Andreas Herzig, Emiliano Lorini, Jomi F Hübner, Jonathan Ben-Naim, Olivier Boissier, Cristiano Castelfranchi, Robert Demolombe, Dominique Longin, Laurent Perrussel and Laurent Vercouter. *Prolegomena for a logic of trust and reputation.* In Proceedings of the 3rd International Workshop on Normative Multiagent Systems (NorMAS2008), pages 143–157. Springer, 2008. (Cited on page 27.)

[Herzig *et al.* 2009] Andrea Herzig, Emiliano Lorini, Jomi F Hübner and Laurent Vercouter. *A logic of trust and reputation.* Logic Journal of the IGPL, vol. 18, no. 1, pages 214–244, 2009. (Cited on page 27.)

[Holland & Melhuish 1999] Owen Holland and Chris Melhuish. *Stigmergy, self-organization, and sorting in collective robotics.* Artificial life, vol. 5, no. 2, pages 173–202, 1999. (Cited on page 64.)

[Huang & Kwiatkowska 2017] Xiaowei Huang and Marta Kwiatkowska. *Reasoning about Cognitive Trust in Stochastic Multiagent Systems.* In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), 2017. (Cited on pages 26 and 27.)

[Huberman & Hogg 1988] Bernardo A Huberman and Tad Hogg. *The Behavior of Computational Ecologies.* In B A Huberman, editor, The Ecology Of Computation, pages 77–115. North Holland, 1988. (Cited on page 10.)

[Huberman & Hogg 1995] Bernardo A Huberman and Tad Hogg. *Distributed Computation as an Economic System.* Journal of Economic Perspectives, vol. 9, no. 1, pages 141–152, 1995. (Cited on page 10.)

[Hübner & Bordini 2006] J.F. Hübner and R.H. Bordini. *Jason*, 2006. `http://jason.sourceforge.net`. (Cited on page 33.)

[Hunkeler *et al.* 2008] U. Hunkeler, H. L. Truong and A. Stanford-Clark. *MQTT - A publish/subscribe protocol for Wireless Sensor Networks*. In Communication Systems Software and Middleware and Workshops (COMSware) 2008. 3rd international conference on, pages 791–798. IEEE, 2008. (Cited on page 32.)

[Jacobi 2013] Ian Campbell Jacobi. *Dynamic application of problem solving strategies : dependency-based flow control*. PhD thesis, MIT EECS, 2013. (Cited on page 25.)

[Jansen 2002] Wayne A Jansen. *Intrusion detection with Mobile Agents*. Computer Communications, vol. 25, no. 15, pages 1392–1401, September 2002. (Cited on page 20.)

[Jennings 2000] Nicholas R Jennings. *On Agent-based Software Engineering*. Artificial Intelligence, vol. 117, no. 2, pages 277–296, 2000. (Cited on page 5.)

[Jøsang *et al.* 2006] Audun Jøsang, Stephen Marsh and Simon Pope. *Exploring Different Types of Trust Propagation*. In Ketil Stølen, WilliamH. Winsborough, Fabio Martinelli and Fabio Massacci, editors, Trust Management, volume 3986 of *Lecture Notes in Computer Science*, pages 179–192. Springer Berlin Heidelberg, 2006. (Cited on page 27.)

[Katis *et al.* 2008] Piergiulio Katis, Nicoletta Sabadini and Robert F C Walters. *On Partita Doppia*. arXiv:0803.2429 [math.CT], 2008. (Cited on page 22.)

[Kmett 2017] Edward Kmett. *Propagators*. Lambda Jam Conference, Brisbane AUS, April 28-29 2016, 2017. (Cited on page 109.)

[Kocherlakota 1998] Naranaya R Kocherlakota. *Money is Memory*. Journal of Economic Theory, vol. 81, no. 2, pages 232–251, 1998. (Cited on page 53.)

[Koster *et al.* 2013] Andrew Koster, Marco Schorlemmer and Jordi Sabater-Mir. *Opening the Black Box of Trust: Reasoning about Trust Models in a BDI Agent*. Journal of Logic and Computation, vol. 23, no. 1, pages 25–58, 2013. (Cited on page 27.)

[Kramdi 2015] Seifeddine Kramdi. *A modal approach to model computational trust*. PhD thesis, Université de Toulouse, 2015. (Cited on page 27.)

[Kuper & Newton 2013] Lindsey Kuper and Ryan R Newton. *LVars: Lattice-based Data Structures for Deterministic Parallelism*. In HPC'13 Proceedings of the 2nd ACM SIGPLAN workshop on Functional high-performance computing - Boston, Massachusetts, USA — September 23 - 23, 2013 , pages 71–84, 2013. (Cited on page 108.)

[Kusmierz 2017] Bartosz Kusmierz. *The first glance at the simulation of the Tangle: discrete model*, November 2017. (Cited on page 98.)

[Kwiatkowska 2013] Marta Kwiatkowska. *Model Checking and Strategy Synthesis for Stochastic Games: From Theory to Practice.* In Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), pages 4:1–4:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. (Cited on pages 23 and 109.)

[Lahijanian & Kwiatkowska 2016] Morteza Lahijanian and Marta Kwiatkowska. *Social Trust: a Major Challenge for the Future of Autonomous Systems.* AAAI Fall Symposium on Cross-Disciplinary Challenges for Autonomous Systems, AAAI, AAAI Press. To appear, 2016. (Cited on pages 8 and 27.)

[Lee 2015] Edward A Lee. *The Past, Present and Future of Cyber-Physical Systems: A Focus on Models.* Sensors, vol. 15, no. 3, pages 4837–4869, February 2015. (Cited on page 7.)

[Leinster 2004] T Leinster. Higher operads, higher categories. London Mathematical Society Lecture Note Series 298. Cambridge University Press, 2004. (Cited on page 22.)

[LeMahieu 2017] Colin LeMahieu. *RaiBlocks: A Feeless Distributed Cryptocurrency Network*, December 2017. (Cited on page 54.)

[Lerman & Spivak 2017] Eugene Lerman and David I Spivak. *An algebra of open continuous time dynamical systems and networks.* arXiv:1602.01017v2 [math.DS], 2017. (Cited on page 22.)

[Lerman 2017] Eugene Lerman. *Networks of Open Systems.* arXiv:1705.04814 [math.OC], 2017. (Cited on page 22.)

[Liggett 1985] T M Liggett. Interacting particle systems. Springer, 1985. (Cited on page 19.)

[Liggett 1999] T M Liggett. Stochastic interacting systems: Contact, voter, and exclusion processes. Springer, 1999. (Cited on page 19.)

[Louzada Pinto 2016] Julio Cesar Louzada Pinto. *Information diffusion and opinion dynamics in social networks.* PhD thesis, Université Pierre et Marie Curie - Paris VI, 2016. (Cited on page 19.)

[Lynch & Tuttle 1989] Nancy A Lynch and M Tuttle. *An introduction to input/output automata.* Quarterly 2(3), Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, September 1989. (Cited on page 21.)

[Lynch 1996] Nancy A Lynch. Distributed algorithms. Morgan Kaufmann, 1996. (Cited on page 21.)

[Mac Lane 1998] Saunders Mac Lane. Categories for the working mathematician. Springer, 2nd édition, 1969,1998. (Cited on page 22.)

[Margolis 2011] Michael Margolis. Arduino cookbook. O'Reilly Media, 2011. (Cited on page 31.)

[Marsh 1994] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994. (Cited on page 26.)

[Matt *et al.* 2010] Paul-Amaury Matt, Maxime Morge and Francesca Toni. *Combining statistics and arguments to compute trust*. In Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), 2010. (Cited on pages 26 and 27.)

[May 1972] May. The geometry of iterated loop spaces. Springer-Verlag, 1972. (Cited on page 22.)

[McKnight & Chervany 2001] D Harrison McKnight and Norman L Chervany. *Trust and Distrust Definitions: One Bite at a Time*. In Rino Falcone, Muhandar Singh and Y H Tan, editors, Trust in Cyber-societies, volume 2246 of *Lecture Notes in Computer Science*, chapter 22-54. Springer Berlin / Heidelberg, 2001. (Cited on page 26.)

[Meiklejohn & Van Rooy 2015] Christopher Meiklejohn and Peter Van Rooy. *Lasp: a Language for Distributed, Coordination-Free Programming*. In Proceedings of PPDP'15, July 14-16, Siena, Italy. ACM, 2015. (Cited on page 109.)

[Miller & Drexler 1988] Mark S Miller and K Eric Drexler. *Markets and Computation: Agoric Open Systems*. In B A Huberman, editor, The Ecology Of Computation. North Holland, 1988. (Cited on page 10.)

[Milner *et al.* 1992] Robin Milner, J G Parrow and D G Walker. *A Calculus of Mobile Processes, I and II*. Information and Computation, vol. 100, no. 1, pages 1–40;41–77, September 1992. (Cited on page 20.)

[Milner 2009] Robin Milner. The space and motion of communicating agents. Cambridge University Press, 2009. (Cited on page 20.)

[Minsky 1986] Marvin L Minsky. The society of mind. Simon & Schuster, 1986. (Cited on pages 9 and 107.)

[Mui 2002] Lik Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks.* PhD thesis, MIT, 2002. (Cited on page 26.)

[Myerson 1991] Roger B Myerson. Game theory: Analysis of conflict. Harvard University Press, 1991. (Cited on page 23.)

[Nakamoto 2008] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*, October 2008. (Cited on page 53.)

[Nash Jr 2002] John Forbes Nash Jr. *Ideal Money.* Southern Economic Journal, vol. 69, no. 1, pages 4–11, July 2002. (Cited on page 53.)

[Newman 2010] Mark Newman. Networks: An introduction. Oxford University Press, 2010. (Cited on pages 39 and 46.)

[Nisan *et al.* 2007] Noam Nisan, Tim Roughgarden, Eva Tardos and Vijay V Vazirani, editors. Algorithmic game theory. Cambridge University Press, 2007. (Cited on page 23.)

[NIST] NIST. *Cyber Physical Systems.* Online in https://www.nist.gov/el/cyber-physical-systems. (Cited on page 6.)

[Nooteboom 2002] Bart Nooteboom. Trust: Forms, foundations, functions, failures and figures. Edward Elgar, 2002. (Cited on page 108.)

[Norretranders 1998] T Norretranders. The user illusion. Penguin, 1998. (Cited on page 10.)

[Norvig & Russell 1999] Peter Norvig and Stuart J Russell. Artificial intelligence: A modern approach. Prentice Hall, 1999. (Cited on page 11.)

[Otteson 2002] James Otteson. Adam Smith's Marketplace of Life. Cambridge University Press, 2002. (Cited on page 9.)

[Paglieri *et al.* 2014] Fabio Paglieri, Cristiano Castelfranchi, Celia da Costa Pereira, Rino Falcone, Andrea Tettamanzi and Serena Villata. *Trusting the messenger because of the message: feedback dynamics from information quality to source evaluation.* Computational and Mathematical Organization Theory, vol. 20, no. 2, pages 176–194, 2014. (Cited on page 27.)

[Palm 1981] Günther Palm. *Towards a Theory of Cell Assemblies.* Biological Cybernetics, vol. 39, pages 181–194, 1981. (Cited on page 11.)

[Paolucci 2013] Mario Paolucci. Agent-based approaches in economic and social complex systems vii, volume 10 of *Agent-Based Social Systems*, chapter Money Emergence on a Network Topology, pages 61–71. Springer, 2013. (Cited on page 107.)

[Papert 1980] Seymour Papert. Mindstorms: Children, computers, and powerful ideas. Basic Books, 1980. (Cited on page 34.)

[Parkes & Wellman 2015] David Parkes and Michael P Wellman. *Economic Reasoning and Artificial Intelligence*. Science, vol. 349, pages 267–272, 2015. (Cited on page 23.)

[Parsons *et al.* 2012] Simon Parsons, Katie Atkinson, Karen Haigh, Karl Levitt, Peter McBurney, Jeff Rowe, Munindar P Singh and Elizabeth Sklar. *Argument Schemes for Reasoning about Trust*. In Proceedings of COMMA 2012, 2012. (Cited on page 27.)

[Patti 2002] Viviana Patti. *Programming Rational Agents: a Modal Approach in a Logic Programming Setting*. PhD thesis, Dipartimento di Informatica — Universitá degli Studi di Torino, 2002. (Cited on page 27.)

[Pinyol & Sabater-Mir 2013] Isaac Pinyol and Jordi Sabater-Mir. *Computational trust and reputation models for open multi-agent systems: a review*. Artificial Intelligence Review, vol. 40, no. 1, pages 1–25, June 2013. (Cited on page 26.)

[Poggiolesi 2011] Francesca Poggiolesi. Gentzen calculi for modal propositional logic. Springer, 2011. (Cited on page 106.)

[Poole *et al.* 2016] Ben Poole, Subhaneil Lahiri, Malthra Raghu, Jascha Sohl-Dickstein and Surya Ganguli. *Exponential expressivity in deep neural networks throught transient chaos*, June 2016. (Cited on page 101.)

[Popov *et al.* 2017] Serguei Popov, Olivia Saa and Eugenio Finardi. *Equilibria in the Tangle*. ArXiV, December 2017. (Cited on pages 58, 59 and 99.)

[Popov 2017] Serguei Popov. *The Tangle*. (v 1.4), November 2017. (Cited on pages 53, 54, 55, 57, 58, 59, 98, 99 and 102.)

[Primiero & Raimondi 2014] Giuseppe Primiero and Franco Raimondi. *A typed natural deduction calculus to reason about secure trust*. In 2014 Twelfth Annual International Conference on Privacy, Security and Trust, Toronto, ON, Canada, July 23-24, 2014, pages 379–382, 2014. (Cited on page 47.)

[Primiero *et al.* 2016] Giuseppe Primiero, Michele Bottone, Franco Raimondi and Jacopo Tagliabue. *Contradictory information flow in networks with trust and distrust*. In 5th International Workshop on Complex Networks and their Applications (COMPLEX NETWORKS 2016), 01-06 Dec 2016, Milan, Italy, 2016. (Cited on pages 46, 47, 49 and 71.)

[Primiero *et al.* 2017] Giuseppe Primiero, Franco Raimondi, Michele Bottone and Jacopo Tagliabue. *Trust and Distrust in Contradictory Information Trasmission.* Applied Network Science, vol. 2, no. 12, pages 1–30, 2017. (Cited on pages 46, 47, 48, 49, 52, 71, 93 and 102.)

[Primiero 2016] Giuseppe Primiero. *A Calculus for Distrust and Mistrust.* In Trust Management X - 10th IFIP WG 11.11 International Conference, IFIPTM 2016, Darmstadt, Germany, July 18-22, 2016, Proceedings, pages 183–190, 2016. (Cited on page 47.)

[Prusinkiewicz *et al.* 1996] P. Prusinkiewicz, A. Lindenmayer, J.S. Hanan and F.D Fracchia. The algorithmic beauty of plants. Springer-Verlag, 1996. (Cited on page 21.)

[Pulvermuller 2002] Friedermann Pulvermuller. The neuroscience of language: On brain circuits of words and serial order. Cambridge University Press, 2002. (Cited on page 11.)

[Quercia *et al.* 2007] Daniele Quercia, Stephen Hailes and Licia Capra. *Lightweight Distributed Trust Propagation.* In Proceedings of the Seventh IEEE International Conference on Data Mining, pages 282–291, 2007. (Cited on page 27.)

[Radul & Sussman 2010] Alexey Radul and Gerald Jay Sussman. *The Art of the Propagator.* Technical report MIT-CSAIL-TR-2009-002, Massachusetts Institute of Technology, 2010. (Cited on pages 23 and 24.)

[Radul 2009] Alexei A. Radul. *Propagation Networks: A Flexible and Expressive Substrate for Computation.* PhD thesis, MIT, 2009. (Cited on pages 23 and 24.)

[Ramirez & Fasli 2017] Wulfrano Arturo Luna Ramirez and Maria Fasli. *Integrating NetLogo and Jason: a Disaster-Rescue Simulation.* In Proceedings of 9th Computer Science and Electronic Engineering Conference (CEEC), 27-29 Sept. 2017, pages 213–218. IEEE, 2017. (Cited on page 102.)

[Rao & Georgeff 1998a] A S Rao and M P Georgeff. *Decision procedures for BDI logics.* Journal of Logic and Computation, vol. 8, no. 3, pages 293–344, 1998. (Cited on page 6.)

[Rao & Georgeff 1998b] A S Rao and M P Georgeff. *Decision procedures for BDI logics.* Journal of Logic and Computation, vol. 8, no. 3, pages 293–344, 1998. (Cited on page 27.)

[Reif *et al.* 2016] W. Reif, G. Anders, H. Seebach, S. Jan-Philipp, E. André, J. Hähner, C. Müller-Schloer and T Ungerer, editors. Trustworthy open self organising open systems. Springer, 2016. (Cited on page 25.)

[Riely & Hennessy 1998] J Riely and M Hennessy. *A typed language for distributed mobile processes.* In Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 378–390, 1998. (Cited on page 20.)

[Robert & Casella 2005] C Robert and R Casella. Markov chain monte carlo in practice. Chapman and Hall, 2005. (Cited on page 54.)

[Rota & Wallstrom 1997] G C Rota and T C Wallstrom. *Stochastic integrals: a combinatorial approach.* Annals of Probability, vol. 25, no. 3, pages 1257–1283, 1997. (Cited on page 108.)

[Rupel & Spivak 2014] Dylan Rupel and David I Spivak. *The operad of temporal wiring diagrams: formalizing a graphical language for discrete-time processes.* arXiv:1307.6894 [math.CT], July 2014. (Cited on page 22.)

[Sainudiin & Welch 2015] Raazesh Sainudiin and David Welch. *The Transmission Process: A Combinatorial Stochastic Process on Binary Trees over the Contact Network of Hosts in an Epidemic.* Technical report, School of Mathematics and Statistics, University of Canterbury, NZ, 2015. (Cited on page 19.)

[Schwartz-Ziv & Tishby 2017] Ravid Schwartz-Ziv and Naftali Tishby. *Opening the black box of Deep Neural Networks via Information*, April 2017. (Cited on page 100.)

[Setter *et al.* 2016] T Setter, A Gasparri and M Egerstedt. *Trust-Based Interactions in Teams of Mobile Agents.* In American Control Conference, Boston MA, 2016. (Cited on page 27.)

[Shanahan 2015] Murray Shanahan. The technological singularity. MIT Press, 2015. (Cited on page 5.)

[Shannon 1953] C E Shannon. *The lattice theory of information.* IEEE Transactions on Information Theory, vol. 1, no. 1, pages 105–107, February 1953. (Cited on page 108.)

[Shoenholz *et al.* 2017] Samuel S Shoenholz, Justin Gilmer, Surya Ganguli and Jascha Sohl-Dickstein. *Deep Information Propagation*, April 2017. (Cited on page 100.)

[Simmel 1907] Georg Simmel. The philosophy of money. Routledge, 1907. (Cited on pages 53 and 107.)

[Singh 2011] Munindar P Singh. *Trust as Dependence: A Logical Approach.* In Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11), pages 863–870, 2011. (Cited on page 27.)

[Sompolinsky *et al.* 2017] Yonathan Sompolinsky, Yoad Lewenberg and Aviv Zohar. *SPECTRE: Serialization of Proof-of-Work Events - confirming Transactions via Recursive Elections*, 2017. (Cited on page 54.)

[Soofi 1994] E S Soofi. *Capturing the intangible concept of information.* Journal of the American Statistical Association, vol. 89, pages 1243–1254, 1994. (Cited on page 10.)

[Soofi 2000] E S Soofi. *Principal Information Theoretic Approaches.* Journal of the American Statistical Association, vol. 95, pages 1349–1353, 2000. (Cited on page 10.)

[Spivak & Tan 2015] David I Spivak and Joshua Z Tan. *Nesting of dynamic systems and mode-dependent networks.* arXiv:1502.07380v3 [math.DS], December 2015. (Cited on page 22.)

[Spivak 2016] David I. Spivak. *The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits.* arXiv:1305.0297 [cs.DB], October 2016. (Cited on page 22.)

[(STOA) 2016] EPRS European Parliamentary Research Service Scientific Foresight Unit (STOA). *Ethical Aspects of Cyber-Physical Systems.* Online in europarl.europa.eu, June 2016. (Cited on page 8.)

[Susnea *et al.* 2009] Ioan Susnea, Grigore Vasiliu, Adrian Filipescu, Adriana Serbencu and Adrian Radaschin. *Virtual pheromones to control mobile robots. A neural network approach.* In 2009 IEEE International Conference on Automation and Logistics, pages 1962–1967. IEEE, 2009. (Cited on page 68.)

[Susnea 2015] Ioan Susnea. *Engineering Human Stigmergy.* International Journal of Computers Communications & Control, vol. 10, no. 3, pages 420–427, 2015. (Cited on page 67.)

[Teacy *et al.* 2006] W.T. Luke Teacy, Jigar Patel, Nicholas R Jennings and Michael Luck. *TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources.* Journal of Autonomous Agents and Multi-Agent Systems, vol. 12, no. 2, pages 183–198, Marchs 2006. (Cited on page 26.)

[Thom 1983] René Thom. Mathematical models of morphogenesis. Wiley, 1983. (Cited on page 10.)

[Tishby & Polani 2010] Naftali Tishby and Daniel Polani. *Information Theory of Decisions and Actions.* In V Cutsuridis, A Hussain and J Taylor, editors, Perception-Action Cycle, Springer Series in Cognitive and Neural Systems. Springer, 2010. (Cited on page 101.)

[Tishby & Polani 2011] Naftali Tishby and Daniel Polani. *Information Theory of Decisions and Actions.* In Cutsuridis V, Hussain A and Taylor J, editors, Perception-Reason-Action Cycle: Models, Architectures and Hardware, Springer Series in Cognitive and Neural Systems, pages 601–636. Springer, New York, NY, 2011. (Cited on page 109.)

[Tisue & Wilensky 2004] Seth Tisue and Uri Wilensky. *NetLogo: A Simple Environment for Modeling Complexity.* Presented at the International Conference on Complex Systems, Boston, May 16–21, 2004. (Cited on page 34.)

[Tsang & Larson 2014] Alan Tsang and Kate Larson. *Opinion Dynamics of Skeptical Agents.* In Alesssio Lomuscio, Paul Scerri, Anna Bazzan and Michael Huhns, editors, Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), pages 277–284, 2014. (Cited on page 26.)

[Vagner *et al.* 2015] Dmitri Vagner, David I Spivak and Eugene Lerman. *Algebras of Open Dynamical Systems on the Operad of Wiring Diagrrams.* Theory & Applications of Categories, vol. 30, no. 51, pages 1793–1822, 2015. (Cited on page 22.)

[van de Bunt *et al.* 2005] Gerhard G. van de Bunt, Rafael P.M. Wittek and Maurits C. de Klepper. *The Evolution of Intra-Organizational Trust Networks: The Case of a German Paper Factory: An Empirical Test of Six Trust Mechanisms.* International Sociology, vol. 20, no. 3, pages 339–369, 2005. (Cited on page 46.)

[Vrba 2013] Pavel Vrba. *Review of Industrial Applications of Multi-agent Technologies.* In Service Orientation in Holonic and Multi Agent Manufacturing and Robotics, volume 472 of *Studies in Computational Intelligence*, chapter 21, pages 327–338. Springer, 2013. (Cited on page 5.)

[Wegner & Goldin 1999] Peter Wegner and Dina Goldin. *Interaction, Computability, and Church's Thesis*, 1999. (Cited on page 21.)

[Weiser & Brown 1996] Mark Weiser and John Seely Brown. *Designing Calm Technology.* PowerGrid Journal, 1996. (Cited on page 20.)

[Weiss 1995] Gerhard Weiss. *Distributed Reinforcement Learning.* Robotics and Autonomous Systems, vol. 15, pages 135–142, 1995. (Cited on page 10.)

[Wellman 1999] Michael P Wellman. *Algorithms fof Decentralised Resource Allocation*, 1999. (Cited on page 10.)

[Wiener 1948] Norbert Wiener. Cybernetics: Or control and communication in the animal and the machine. MIT Press, 1948. (Cited on page 7.)

[Wikipedia 2018a] Wikipedia. *Adjacency List*, 2018. (Cited on page 55.)

[Wikipedia 2018b] Wikipedia. *Diffusion-Limited Aggregation*, 2018. (Cited on page 77.)

[Wilensky 1999] Uri Wilensky. *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, Illinois, 1999. (Cited on page 34.)

[Wooldridge 2003] Michael A Wooldridge. An Introduction to MultiAgent Systems. John Wiley & Sons, 2003. (Cited on pages 5 and 6.)

[Wright & Aubé 1998] Ian Wright and Michel Aubé. *The Society of Mind Requires an Economy of Mind.* http://www.cs.bham.ac.uk/research/projects/cogaff/Wright_ Aube_ eom.pdf, 1998. (Cited on page 10.)

[Yamada & Abramsky 2017] Norihiro Yamada and Samson Abramsky. *Dynamic Games and Strategies.* arXiv:1601.04147v3 [cs.LO], January 2017. (Cited on page 23.)

[Yamada 2017] Norihiro Yamada. *Game-theoretic Model of Computation.* arXiv:1702.05073v2 [cs.LO], 2017. (Cited on page 23.)

[Yasutomi 1995] Ayumu Yasutomi. *The Emergence and Collapse of Money.* Physica D: Nonlinear Phenomena, vol. 82, no. 1-2, pages 180–194, April 1995. (Cited on pages 53 and 107.)

[Yau 2017] Donald Yau. *Operad of Wiring Diagrams.* arXiv:1512.01602 [math.CT], 2017. (Cited on page 22.)

[Zenil *et al.* 2017] Hector Zenil, Angelika Schmidt and Jesper Tegnér. *Causality, Information and Biological Computation: An algorithmic software approach to life, disease and the immune system.* arXiv:1508.06538v5, 2017. (Cited on page 10.)

[Ziegler & Laursen 2005] Cai-Nicolas Ziegler and Georg Laursen. *Propagation Models for Trust and Distrust in Social Networks.* Information Systems Frontiers, vol. 7, no. 4:5, pages 337–358, 2005. (Cited on page 27.)