Huib Aldewereld
Jaime Simão Sichman (Eds.)

LNAI 7756

Coordination, Organizations, Institutions, and Norms in Agent Systems VIII

14th International Workshop, COIN 2012
Held Co-located with AAMAS 2012, Valencia, Spain, June 2012
Revised Selected Papers

Springer

Huib Aldewereld Jaime Simão Sichman (Eds.)

Coordination, Organizations, Institutions, and Norms in Agent Systems VIII

14th International Workshop, COIN 2012
Held Co-located with AAMAS 2012
Valencia, Spain, June 5, 2012
Revised Selected Papers

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Huib Aldewereld
Delft University of Technology
Department of Technology, Policy and Management
Section ICT
Jaffalaan 5, P.O. Box 5015
2600 GA Delft, The Netherlands
E-mail: h.m.aldewereld@tudelft.nl

Jaime Simão Sichman
Universidade de São Paulo
Av Prof. Luciano Gualberto
158 travessa 3
05508-970 São Paulo, SP, Brazil
E-mail: jaime.sichman@poli.usp.br

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-37755-6 e-ISBN 978-3-642-37756-3
DOI 10.1007/978-3-642-37756-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013935398

CR Subject Classification (1998): I.2.11, I.2.0, I.2.3, I.2.6, I.2.8, H.3.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The 2012 edition of the international workshop on Coordination, Organization, Institutions and Norms was the 14^{th} occurrence of a series that began in 2006, as may be found at *http://www.pcs.usp.br/∼coin*. Moreover, it was the 7^{th}occurence of the workshop within the AAMAS conference.

All these years, the workshop has expressed that coordination, organizations, institutions, and norms are four key governance elements for the regulation of open multi-agent systems, and this year was no different. The workshop tried to constitute a space for the debate and exploration of these four elements that are central in the design and use of open systems. The submitted papers all touch on one (or more) of these main topics of the COIN workshops.

COIN 2012 was hosted at the 11^{th} International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), which was held in Valencia, Spain, during June 4–8, 2012. In this edition, 28 submissions from 11 countries were received, from which we selected 13 papers for presentation at the workshop (near 45% acceptance).

2012 was a year of changes. COIN used to have a (rotating) Steering Committee that ensured the quality of the workshop. During the yearly COIN Committee meeting in Valencia it was decided that it was time for a new form of leadership: COIN is now a community-driven workshop. The other change for COIN was that there was only one single COIN event this year, the workshop co-located at AAMAS 2012. Hence, this volume contains 10 out of 13 revised versions of papers presented at the workshop. We want to acknowledge the quality of the reviews of both the selection of papers for the workshop as well as the reviewing for the proceedings. We would like to thank the PC members for their time and effort in reviewing the papers in both rounds, which have made this volume of the quality level expected of a COIN proceedings.

We are also grateful to all the authors; the high quality of the submitted papers made our choice of acceptance a difficult one. With the amount of good-quality submissions, it was a hard decision to reject some of the papers, but unfortunately we could not fit more into the workshop program. While the program was rather packed, there was still ample room for lively discussion. The high level of the discussions clearly had its impact on the quality of the revised versions of the papers, presented in this volume.

Finally, thanks are due to Elisabeth Sklar (AAMAS 2012 Workshop Chair), to Wiebe van der Hoek and Lin Padgham (AAMAS 2012 General Co-chairs), and to Vicente Botti (AAMAS 2012 Local Organising Chair) for the wonderful ambience in Valencia.

January 2013

Huib Aldewereld
Jaime Sichman

Organization

General and Program Chairs

Huib Aldewereld	Delft University of Technology, The Netherlands
Jaime Sichman	University of Sao Paulo, Brazil

Program Committee

Sergio Alvarez-Napagao	Universitat Politècnica de Catalunya, Spain
Alexander Artikis	NCSR “Demokritos”, Greece
Guido Boella	University of Turin, Italy
Olivier Boissier	ENS Mines Saint-Etienne, France
Patrice Caire	University of Namur, Belgium
Cristiano Castelfranchi	Inst. of Cognitive Sciences and Technologies, Italy
Antonio Costa	Universidade Federal do Rio Grande, Brazil
Luciano Coutinho	Universidade Federal do Maranhão (UFMA), Brazil
Marina De Vos	University of Bath, UK
Gennaro Di Tosto	Utrecht University, The Netherlands
Virginia Dignum	Delft University of Technology, The Netherlands
Nicoletta Fornara	Universita della Svizzera Italiana, Lugano, Switzerland
Chris Haynes	King’s College London, UK
Jomi Fred Hubner	Federal University of Santa Catarina, Brazil
Joris Hulstijn	Delft University of Technology, The Netherlands
Eric Matson	Purdue University, USA
John-Jules Meyer	Utrecht University, The Netherlands
Simon Miles	King’s College London, UK
Pablo Noriega	IIIA-CSIC, Spain
Eugénio Oliveira	Universidade do Porto, Portugal
Andrea Omicini	Università di Bologna, Italy
Sascha Ossowski	University Rey Juan Carlos, Spain
Julian Padget	University of Bath, UK
Alessandro Ricci	University of Bologna, Italy
Juan Antonio Rodriguez Aguilar	IIIA-CSIC, Spain

Additional Reviewers

Table of Contents

Situating COIN in the Cloud
(Invited Paper)

Julian Padget

University of Bath, Dept. of Computer Science
jap@cs.bath.ac.uk

Abstract. We start from the view that the central theme of the research at the core of coordination, organization, institutions and norms, is whether the social structures and mechanisms, that have emerged over time, can be adapted and applied to artificial societies of programs and perhaps more significantly, to mixed societies of humans *and* programs –and how The means by which the social constraints that guide and regulate behaviour are acquired and represented remains an open problem. If recent experiences in information retrieval and natural language processing are plausible indicators, the statistical may yet oust the logical. Technology aside, it is clear that for socio-technical systems, that integrate human and software components, we may expect the adoption of, or the illusion of observation of, and support for human social conventions. The growing migration to cloud computing of the services that make up current pervasive, social applications suggests near-term developments emerging from the same platform(s). Thus, the question considered here is, what pathways, opportunities and challenges exist for the development, wider use and validation of COIN technologies to help realize socio-technical systems that better meet human requirements. As examples of specific enabling technologies, we review current developments in resource-oriented architecture, complex event processing and stream reasoning and observe how COIN technologies might integrate with them.

1 Introduction

At the first meeting of the COIN series in 2005, Noriega [36] spoke of “fencing the open fields” as an analogy for the need to provide suitable regulation of participant interactions in open systems governed by electronic institutions. However, the risk is that designers, in their desire to ensure the “right” outcomes, may lock actors into a protocol strait-jacket, simply because it is then possible to prove through brute-force search that compliance is assured and only good results follow. In seeking to formalize electronic institutions, this writer fell into the trap of over-specification, attempting to use the π-calculus to describe elements of the FishMarket [19]. We may observe similar trends in both policy- and law-making, where in some cases the political debate appears to exhibit a strong desire to circumscribe individual actions for the sake of the goal of achieving high levels of compliance. Paradoxically, such approaches would appear to be counter to the underlying principles, at the formal end, of game theory, or at the informal end, of economics and the notion of free markets and the ‘invisible hand’. In each case, it is precisely freedom of action that imbues the arena with the flexibility to adapt to changing circumstances and to provide the essential space – 'wiggle room', if you will – to explore as well as exploit [30]. Thus, optimization – interpreted as a high-level of compliance, regardless of cost – may be preferred to satisficing [45]. Costs here may be reflected in a lack of agility to respond to change as much as in the actual cost of policing [9].

We may also consider the freedom to move from a genuinely architectural perspective – a theme we pick up later with reference to patterns – by recalling that Alexander [3] sought what he called "quality without a name" in the design of habitable spaces. Effective, by some definition of the word, social institutions, whether they capture policies or laws, or even social conventions, can equally be viewed as habitable spaces: desirable properties they may typically exhibit are flow, ease of use and low overheads (action, cognition). There has inevitably been a reaction[1] to the up-take of design patterns and the consequences of it, because while Alexander's vision was that inhabitants would design and build their own environment, that is largely unrealistic for most users of software. The situation is potentially different however in respect of normative frameworks – if we are prepared to give participants – both human and software agents – the power to change the rules that govern them and prepared to build the tools to accommodate that change while maintaining system integrity [7,8].

The challenge is how to regulate loosely enough to provide sufficient autonomy and flexibility, but tightly enough to meet system objectives, where most, rather than all, participants behave correctly. Indeed, such a notion of optimization may be illusory, in that all it ensures is behaviour as foreseen by the designer in advance of use. The software engineering literature and the security literature provide many anecdotes on the subject of how system integrity can be undermined by actual usage, not for any malicious motive, but simply because practice discovers more efficient – again, by some definition of the term – ways to achieve the same goals. This observation is central to the argument that underpins process mining, which seeks to discover the so-called 'desire lines' (also called more popularly elephant path, social trail, goat track or bootleg trail) that indicate the sequences of events that occur in practice. This is in contrast to the view of business process engineering, which sees such sequences as almost certainly non-optimal from a business point of view (cost, speed, reliability) and sees instead meanderings of little consequence which they call 'cow paths' [1]. Thus, whether the path is due to an elephant – and is worth preserving – or a cow – and should be eradicated – seems to depend on point of view.

The consensus in normative systems research might now appear to be in favour of regulation (good?) rather than regimentation (bad?). But even that may not be a permanently tenable position, because while the balance between the two can be established through *a priori* design choices, it may be desirable to adjust the degree of autonomy subject to prevailing conditions. These conditions may be reflected in system metrics, leading to a shift from regulation towards regimentation or vice versa, as circumstances dictate.

Let us now consider some more pragmatic issues, such as: (i) where shall the many software components execute that comprise such systems, (ii) how shall data sent between them be represented, and (iii) how shall communications between them be

[1] There are many reports on, but no definitive link to, the Gang of Four trial at OOPSLA'99.

achieved? These are not new questions for the agent community, although unfortunately our attempts to reach a conclusion on at least the latter two have been unsuccessful. These questions are not new, either, for the wider computer science community. The momentum that is forming around cloud computing not only suggests an answer to the first issue, but also that we should consider establishing common ground with the accompanying technologies of web services, publish/subscribe communications protocols and semantic annotation. Consequently, COIN technologies can be part of, rather than apart from, the cloud environment.

To return to the issue cited at the opening of fencing the open fields, the challenge left for the community then was "how to put the bell on the cat". Put another way, if we believe normative specifications and norm-aware actors are the answer, how are we to get these features into open systems, the designers of which may well see no need for such bells[2]. The purpose of this somewhat wide-ranging introduction has been to make connections with a variety of research and practice, inside and outside computer science to try to provide some form of backdrop for the current state of COIN technology research. We review the state of COIN research in the next section and put forward some of the significant tasks that we believe face us. Then in section 3, we examine the features of cloud computing that together offer an excellent experimentation and evaluation environment. Using these we can show what COIN technologies might achieve, so that we may indeed "bell the cat". We conclude with some suggestions for actions that the COIN community might take to initiate the transition to cloud computing and at the same time create greater synergy across the community.

2 The State of COIN Technologies

The aim of this section is to examine from a high level what has been achieved in COIN technologies over the last decade and a half, but also to identify the various issues that stand in the way of the wider recognition of the utility it offers and, more significantly, what needs to be done to provide externally acceptable validation of the technologies. Several factors are put forward as problems relating to COIN technologies, but the overriding issues are connectivity – how to join our tools and components with the wider software world – and usability – how to join our concepts and approaches with the wider software community. The citations are intended as representative of relevant work rather than as an exhaustive survey. A complete list of the COIN volumes can be found at `http://www.pcs.usp.br/~coin/`.

Over a period of more than 15 years, COIN technologies have evolved from static normative frameworks, encoded implicitly in the control logic of software components, through explicit representations regimenting agent behaviour in trading platforms [43] or informing agent choices in agent-based simulation [34], to guiding agent and human actions in complex mixed environments [35]. But while this progression demonstrates real advances in norm representation and reasoning, those demonstrations are largely limited to small, carefully constructed illustrative cases – not necessarily helped by the publication format of the conferences and journals that the community uses, as well as

[2] or whistles.

other academic environment factors – rather than showing impact on large scale systems driven by real, rather than synthetic, data.

There is little originality in the following observations on the criticisms that can be levelled at COIN technologies and could equally be applied to other areas of computer science, or even science in general:

1. **Plausibility:** this is the first barrier to up-take. We have no demonstrators that make the case in practice for COIN technologies; there is only the potential and the case is not compelling because existing systems work – or appear to – on small and carefully selected illustrative scenarios.
2. **Scalability:** part of the lack of plausibility stems from a lack of evidence either in the form of deployed systems or from theoretical analysis of the capacity for the technologies to scale; we need to demonstrate solutions at scale. But this is a chicken-and-egg situation, because we also need host systems that can be augmented post-facto by COIN technologies in order to be able to make that case.
3. **Visibility:** the benefits of the use of COIN technologies need to be clear and to offer substantive improvement, possibly in several ways, over conventional technologies. But, frustratingly, the best indicator of effective application of COIN technologies may be that they are barely noticeable.
4. **Packaging:** a practical barrier to up-take depends upon how COIN technologies are delivered. New technologies that either require re-training, new interfaces or discarding (part of) the existing software base inevitably face greater resistance to up-take than something that integrates by means of widely-used interfaces and which enhance rather than replace. Not least, the capacity for turning enhancements on and off may provide a valuable way to demonstrate their impact. We need to deliver COIN technologies in packaging that not only helps us as a community to integrate and evaluate what we do, but also to integrate with minimal overhead or impact with legacy systems.

COIN technologies have the potential to contribute to the creation of systems that are all of open, distributed, intelligent and adaptable – even if those terms might require an essay each of their own to circumscribe expectations and establish the connections between them. But the software we have built so far typically has limited *interoperability*: a Java library can be a useful component and might with some effort be deployed as a web service, but lack of systems experience means it is hard to say whether an API is a suitable interface or whether a richer communication language is desirable. A further artefact of the development process is the difficulty of *re-usability*: although the technologies aim to be and often are quite general purpose in nature, the supporting software can be quite sensitive to deployment environment and hard to maintain even in the short to medium term. *Performance* is also often over-looked, not least because the development scenarios that illustrate the properties we wish to demonstrate (for academic purposes) are quite small, but also because it is hard to identify useful metrics and because the decision procedures in play do not, or cannot, have well-defined performance profiles. Much COIN technology software is written to demonstrate that a particular behaviour or envelope of behaviours can be realized, but there is little culture as yet in the practice of *patterns* for COIN technologies: this may percolate through from the underlying software base, but that seems more likely to address how some function is realized, rather than the function itself. Finally in this tally of criticisms, there is the matter of *resilience*: given that flexibility and adaptation in the face of changing circumstances are among the benefits that should follow from COIN technologies, it seems essential that it should be possible to demonstrate such properties in our own software – that is, reflection – and not just in the systems to which it is applied.

2.1 The Agent View

Previously in intelligent agent research, agent architecture was a major topic from the earliest days of agent-based simulation (ROSS, SWIRL and TWIRL [32]) and subsequently with the more complex layered architectures such as Touring Machines [20] and InteRRap [33]. However, these are now largely forgotten and either regimented agents, such as in e-Institutions [43] and $\mathcal{M}$OISE^{+} [26], or BDI and variants appear to be the common choices. In consequence, a number of those variants of BDI have sought to address the matter of how an agent could and should avail itself of normative reasoning. This has lead to a dichotomy between the internal approach, which further divides between full incorporation [4] and separation of BDI and normative knowledge [14] and the external approach, where normative positions are communicated to agents in the form of obligations [2], determined by normative reasoning components [13].

The convergence of architectural choice on BDI makes system comparison somewhat simpler, but has also had the effect of pushing processing into the agent that is not necessarily so easily handled at that level. Specifically, complex trigger formulae for BDI actions are both hard to test, inasmuch as BDI testing is feasible [48], and to maintain. Furthermore, this hard-codes plan triggers into agents, reducing scope for resilience, since they cannot easily be adjusted in response to changing circumstances.

2.2 The Organization View

In parallel with the evolution of agent architecture, the twin notions of institution and organization have also undergone significant development. In the first instance, there was the FishMarket [44], in which the agents cede control to a governor that directs which actions they shall take in order that each agent be fully compliant with the rules of the institution. Although the approach is somewhat different, $\mathcal{M}$OISE^{+} achieves similar goals, in which agents are constrained by the role they play within a group, so that agents are in effect regimented. A second group of approaches to institutional specification have favoured agent autonomy over guaranteed compliance, using a variety of formalisms [22,12,13,46], in which the common trait is an external entity that reasons about agent actions in respect of the governing norms and identifies normative positions and obligations acquired by agents in consequence.

Dignum and Padget [15] put forward a view that brings together organization and institution. Here, the latter capture the regulations pertaining to specific contexts, and the former expresses the combination of the many institutions that together describe the processes that make up an organization together with the roles of the actors that participate in those institutions.

3 COIN in the Cloud

The preceding sections have summarised a view on the current state of COIN technologies and also put forward some shortcomings that we believe must be addressed in order to raise awareness of the technologies outside the immediate agent community. The purpose of this section is to assess how COIN and cloud technologies might fit together and what actions we might take as a community to bring that about.

The most significant and attractive feature of cloud computing is that it offers provision on demand. Furthermore, that provision can be configured through virtual machines to exactly the combination of operating system and resources that a particular program requires, facilitating the deployment of legacy codes so they can be accessible from anywhere. Cloud computing is also unavoidably distributed, which makes it essential that we establish some conventions on how to interact with COIN deployments – some approaches are discussed below. Distribution offers the opportunity not only for the community to share software (as a service), but also to take advantage of the mechanisms such as enterprise bus architectures, distributed messaging systems and different forms of web services to connect our components with one another and with a huge range of other services. This could potentially significantly reduce our development burden by re-using rather than re-inventing.

Cloud computing can also be seen as the product of an evolutionary process in computing systems. This began with closed systems in which all the components are the product of a single team running on one computer, moved through the (re-)use of libraries and components connected by CORBA running on several computers (on an intranet) and now seeks the creation of increasingly open systems. In this last, components may be replaced/upgraded in live deployments over a range of platforms and edge devices across the internet.

Clearly, from a systems management perspective, the complexity increases significantly in the transition from one computer, to many on a LAN, to many – where devices join and leave the system over time – on a WAN/internet. The governance of such systems has to be distributed, with many components making decisions on the basis of local circumstances, but also, crucially, informed by guidelines for the perceived correct running of the system, which is where COIN technologies are key. Recognition that both much data is being created all the time and that its rapid interpretation is necessary (until we establish what is actually worth collecting) forms the motivation for the SHINE project[3], which brings together sensors, social media and intelligent agents to answer complex information retrieval requests in real-time. But not only are such systems conduits for data and information for delivery to human users, they are also huge data sources in their own right, as each component can be viewed as a monitoring element that feeds into an over-arching process that can observe system health, identify the early stages of anomalous or undesirable situations and alert system control agents of the need to consider whether action need be taken and what it might be [16,17].

Another inevitability concerns the relationship between the producer and consumer of such data. Up to a certain level, a request-response communication model, as has conventionally been used in agent systems. This follows the tradition of procedural

[3] `http://direct.tudelft.nl/shine-117.html`

programming, through remote procedure call to SOAP-based web services – and can work, as long as network latency is low enough. However, asynchronous communications in the form of event notifications, publish/subscribe and RESTful web services are seeing increasing adoption as mechanisms that, while imposing constraints on the programming model, manage to insulate components from the innate and unpredictable latencies in operating across wide area networks. This perspective leads to a view of software components – of all kinds – as a rich variety of event processors – both as consumers and producers of events – connected together in loose and dynamic opportunistic networks.

Unfortunately, although by design, producers know nothing of the requirements or capabilities of consumers and the data volumes can be hard to process and assimilate, if they are not at the information level and frequency commensurate with the consumer's reasoning cycle. For example, a BDI reasoning cycle in one Jason application (driving autonomous vehicles in simulation) [31] appears to be able to cope with percept updates every few hundreds of milliseconds, although this frequency will depend upon number of plans, the complexity of their triggers and the duration of their actions. The critical issue here is the rate at which a consumer can process the event streams it is receiving in order for it to be able to achieve its intended level of situational awareness. This in turn can be exacerbated by the level of the event information being incompatible with the plan triggers (for example) – typically by being at a lower level – so that patterns comprising several events, and possibly several event sources must be recognized [41] before a plan can be triggered. Such processing is typically difficult to develop, debug and maintain *within* an agent framework and would be better out-sourced to a component that can deliver a single percept covering a set of events that characterize a given situation. This leads to the creation of more independent software components in an increasingly complex communication network and a consequent need for further system health monitoring and appropriate presentation of that data, if there is to be any chance of identifying anomalous behaviours.

Looking back at the above, there appear to be several aspects of cloud computing that have features that we need both to demonstrate the benefits of COIN technologies, and also to provide the facilities we need to demonstrate those benefits, amongst which:

1. Sources of data both about the system and the application domain
2. The capacity to create (on demand) the computational facilities to interpret it
3. Means to deploy legacy code in bespoke environments through cloud-based VMs
4. On demand creation of new services to filter, aggregate and summarize data comprising multiple events, even from multiple sources.

3.1 Understanding the Situation

If all the above can be realized, it puts us on a path towards the creation of system that can construct degrees of situational awareness, both about the state of their own system (self-awareness) and awareness of situations as they evolve in the application domain of the system. The outcome should be the means to monitor and to take action on both system health and actor health. Eventually, it should even become feasible to discern the desire lines [1] and, following some collective decision procedure, implement a consistent revision of the governing norms [8].

We want to enable software agents to make good and timely action choices in a variety of situations, thinking specifically about those created by: (i) virtual environments, where there may be a mix of software- and human-controlled avatars and (ii) (imagined) socio-technical systems, such as in search and rescue, military, emergency-response and medical domains.

Intelligent behaviour, or behaviour that is perceived as intelligent, may be attributable to many factors. The one of concern here is how an agent uses information about its situation to make choices of actions. The right choice may be regarded as intelligent. The wrong choice and at worst the agent or its collaboration partners, or even the humans for which it is working, may be exposed to some risk. The situation may not be as black-and-white as just described, but rather there may be better choices, amongst which a dominating choice may not be readily apparent, and worse choices. Optimisation is not required, but satisficing is. Seemingly less serious, at first sight, is that by making a worse choice, trust and plausibility is forfeited and the agent is seen as just another program. However, this too can be critical, not because some pretence need be maintained, but because the loss of reputation and believability induces distraction in a human participant, since they are now looking for the next "mistake", as well as everything else they are doing. As a result, the whole collaborative activity may no longer achieve its goals.

Although there are many factors affecting the effectiveness of such mixed system, we highlight two that we believe have a significant influence on everything else: speed of response and breadth of knowledge. A third aspect is a corollary to these two: the matter the communication of data between components.

Speed of Response. Early agent architectures, such as those mentioned earlier [20,33] sought to reflect then current AI thinking, influenced by research in robotics, by proposing a layered architecture comprising reactive, deliberative and generative components, reflected in the apotheosis of this line of development by Soar [29]. Meanwhile, as noted earlier, the agent community has largely converged on the BDI architecture to fulfil the function of the deliberative and generative layers. In consequence, the reactive layer is effectively subsumed into BDI as events lead to the addition of percepts that in turn cause actions. This usefully simplifies the agent structure and programming because control is all expressed through some variant of the AgentSpeak language. However, the BDI architecture is not designed for the rapid assimilation and assessment of high frequency data and even if higher performance implementations were available, the relative sophistication of the architecture is at odds with the task asked of it. Since the purpose of BDI is to support the deliberative component of an agent, this suggests that: (i) high frequency data needs to be processed and somehow summarized to provide lower frequency data with an implied higher informational value, and (ii) such processing must use a lower overhead computational model to accommodate the high frequency data rate. Out-sourcing this task to a process situated between the data source and the agent, therefore seems sensible. Connectivity can be addressed through some of the distributed computing technologies identified earlier.

Breadth of Knowledge. Understanding the situation should allow us to choose an appropriate action both for the situation and with respect to individual and group goals. Programming an agent to be prepared for any and every situation is clearly infeasible.

Thus, while decision-making must remain the responsibility of the agent, the information upon which that decision is the result of the assessment of sensed data from a variety of sources across a range of time frames and representations. Again, it is likely to be infeasible and undesirable to integrate all such assessment processes within an agent. Thus, although the BDI agent has the capacity for deliberation, not all deliberations are within its capabilities and in common with both software engineering and societal structures, such deliberation could be delegated, leading to recommendations from which to choose. From this, again two conclusions follow: (i) specialized domains may be better reasoned about externally, producing summary recommendations for the agent to choose between and (ii) such processing can use representations and resources appropriate to the domain, or indeed re-use existing reasoning systems. Once again, the conclusion from this is the desirability of out-sourcing the task to a process that may function more like a service, collecting inputs from several sources and publishing results either when ready or on request. A particular case in point are the institutions that comprise an organization, for example, where it is the institutions that act as repositories of social state(context) and provide the function of social reasoning to identify violations and obligations. It nonetheless remains the responsibility of agents to decide what actions to take, in the light of information received from the institutions.

Making Connections. There are two issues to address under the heading of connectivity: (i) how to pass data between the components – representation and protocol – and (ii) how to package existing software to operate in such an environment, and each will affect the other. Our aim is for a low overhead, low maintenance connection fabric, preferably that can be relied upon for support in the wider internet community for some time. Network speeds continue to increase, but latency, as a relative factor, does not change significantly as a proportion of the delay. Thus, although it is traditional to speak of synchronous and asynchronous systems, the practice reflecting the physical constraints of the internet, is either for loosely synchronizing systems, where one side may pause or continue working while waiting for a response from the other side, or asynchronous systems in which components push out data without concern for the receiver, while other components pull in data as it suits them.

Much effort has been expended in the agent community on trying to reach agreement on what and how to communicate between agents. The results have been inconclusive and furthermore are unlikely to see up-take, or indeed support, outside the agents community. Therefore, we suggest that the pragmatic solution is to adopt widely used, maintained and developed standards, so that the task for the community is just to track those standards and build components that utilise them. Specifically, for protocols, this suggests something based on HTTP, to ensure traffic across firewalls, such as the eXtended Messaging and Presence Protocol (XMPP) and for content, something based on RDF, but possibly defined in terms of OWL in order to put constraints on the RDF.

Having discussed broadly the form of some requirements, the following two sections present brief introductions and some indicative references to technologies that may offer some solutions. These are resource-oriented architecture and event processing. The rationale for highlighting these two is firstly, their fit with the intrinsic computational properties of the internet as a distributed computing environment and secondly, as emerging maturing technologies and frameworks, into which we can embed and package COIN technologies, in order to deliver reach beyond individual research groups and beyond the COIN community.

4 Resource Oriented Architecture

It is perhaps an oversimplification to say that ROA is for RESTful web services [47] what Service Oriented Architecture (SOA) is for RPC-style (SOAP) web services, but it does capture the sense of the relationship, in that REST should be seen as one way of achieving resource orientation [38]. In practice, ROA, as do events, enables decoupling of components based on stateless message exchange. Stateless however does not mean state cannot be modelled, but rather that each state, if so required, is a new resource, identified by a new, unique URI. Furthermore, message exchange does not mean request-response in the RPC sense, rather an operation (the request) typically results in a new resource, identified by URI (the response). From this perspective, it has much in common with functional programming.

Although Resource Oriented Architecture denotes a general purpose set of principles in respect of web application design, it is fair to say that its synthesis has been driven by the concrete aspects of RESTful web services and the principles of addressability, statelessness, connectedness, and a uniform interface [42]. Consequently,

1. Resources can be universally identified by unique addresses (addressability),
2. Every request to a resource should contain all the information needed for further processing (statelessness),
3. A resource representation should contain the addresses of all related resources (connectedness) and
4. All resources can be manipulated through uniform methods (uniform interface).

REST-compliant Web services differ from RPC-style Web services in the protocol employed between client and server, in that the latter requires each application designer to define a new and arbitrary protocol comprising vocabulary of nouns and verbs that is usually overlaid on the HTTP protocol[39]. In contrast, REST services work directly with the HTTP verbs of POST, PUT, GET and DELETE which map to the four basic functions of data storage, namely Create, Update, Retrieve and Delete.

As with adopting a purely event-oriented approach, ROA/REST puts constraints on design (and implementation), which can initially be tiresome. But, as with functional, or perhaps a better analogy would be with dataflow/single assignment languages, there are potentially significant benefits, when operating in a distributed environment and especially one with unpredictable levels of latency. Specifically cited benefits [21] include simpler protocols, better synergy with underlying web components, and lower application design costs, but it is hard to find studies that substantiate these largely qualitative claims.

5 Event Processors

This section discusses how to carry out data analysis on behalf of agents, based on the twin criteria of speed of response and breadth of knowledge, identified above. We review some of the work in this area over the past few decades and evaluate its suitability for incorporation into the connection fabric outlined earlier.

5.1 Real-Time Expert Systems

The 1970s and 80s saw the development and refinement of expert systems and the shells used to author them. Two directions emerged from this activity that are still in use today: (i) rule-based systems utilising the RETE algorithm, and (ii) the Prolog language, which although not solely used for building expert systems, nevertheless provides a conceptually similar environment, where facts drive inferencing, expressed through rules. A handful of tools now represent the first group, with some commercial (Ilog Rules) and a few free/open-source (JBoss Rules, JESS) examples. Most recent publications are domain-oriented, focussing on how an expert system has been applied in a particular control context. This underlines the maturity and stability of the underlying technology, which is either RETE (the net algorithm) or Selective Linear Definite clause resolution (logic languages).

A naive implementation of a rule selection process is linear in the number of rules. Any given rule set is finite and so has an upper bound on the matching time required, but is not an adequate guarantee of real-time response, because it means that the number of rules and the complexity of the rule conditions that determine which are applicable must be constrained in order to meet performance requirements. RETE effectively compiles the left hand sides of the rules to build a decision network that identifies the conflict set (of rules that match the current state). While, this and subsequent improvements, are better than the naive approach, complexity is still $O(n)$. The story is broadly the same in logic languages, but with semantic changes (committed choice) and the use of parallel resolution algorithms helping to improve performance.

The key requirement is to be able to know how long the match process will take. This in turn can be affected by placing limitations on the left-hand side conditions resulting in the network having particular time-based properties. The need for real-time response was recognized decades ago and some of the issues surrounding the delivery of such performance are addressed in [28], while a notable contemporary example of the application of the technology was IBMs YES/MVS system [18]. A current example is GENSYMs G2[4], which appears to have significant up-take in process control settings.

RTES provide general programmability through a declarative framework as well as the means to accumulate data over arbitrary long time windows. The programmability also affects how real-time a particular system can be in practice. Prediction of match time can be computed off-line (static analysis of the rules). However, apart from providing an upper bound on the cycle time, there appears to be no way either to guide or to constrain the programmer to produce a match procedure with guaranteed performance. It may be possible to apply syntactic restrictions to the rules so that the (RETE) networks generated are of limited propagation depth. But either way – syntactic restriction or cycle time bound – is not a solution if it means expressing the reasoning logic becomes either contorted, creating a maintenance problem, or it is just impossible to express what is wanted. Furthermore, the match procedure is very fragile in respect of the system as a whole, because it can be pushed outside its performance envelope either by accommodating new requirements or by an increase in the data rate of one of its subscription feeds.

[4] Retrieved from `http://www.gensym.com/`, 20130111

5.2 Complex Event Processing (CEP)

Like many concepts in Computer Science, event processing is not new and could, terminologically, be traced back to the earliest operating systems and soon after with the development of discrete event simulation frameworks such as GPSS [24] and Simula [37]. But there, events were the drivers (of simulation), rather than the subject of analysis themselves. The emergence of the topic of verification as a topic within the design process of various kinds of systems, at both hardware and software levels, focussed on the scrutiny of event traces to detect desirable or undesirable patterns of behaviour. Model checking languages and the tools associated with the various calculi of concurrent systems are some of the computational approaches that have resulted from the objective of understanding large collections of traces in their entirety. Such systems analyse all possible systems states – and the paths between them – in pursuit of the establishment of system invariants. In contrast, ad-hoc solutions to analysing such things as packets on networks, transactions in distributed systems and intrusion detection systems laid the foundations for looking at fragments of traces for significant or anomalous activity. Financial markets were early adopters of the conceptual model, using it to process real-time market feeds for events of significance.

The languages for expressing CEP have become more sophisticated and now offer functions to filter, correlate and aggregate data. The practical question remains however, of how quickly can or must such operations be carried out. Esper [5] and Drools Fusion[6] are typical examples of event processing engines – which notably have a strong commercial orientation, coupled with up-take in the financial sector. The conceptual model is that the user registers queries with the event processor, which will then invoke the query when its conditions match. In Esper, the matching conditions can express durations, the composition of several different streams, filtering, aggregation and sorting. An important abstraction feature is the means to glue together statements using "followed by" conditions. The summary of features that follows is abstracted from the Esper tutorial[7]. Esper's programming language shares some syntactic features with Structured Query Language (SQL) for accessing relational databases, in particular in its `select` and `where` clauses, but the operations are carried out on views – finite length fragments – over streams rather than tables. Views can represent windows over a stream of events, specified by time or count. Views can also sort events, derive statistics from event properties, group events or handle unique event property values. Furthermore, a window can be used to preserve data, in effect by defining an internal table that can be used for input or output by other queries. It is not clear how performance is guaranteed, but the finiteness of the windows and the relative simplicity of the queries that can be performed would certainly limit demand for processing time.

[5] Retrieved from `http://esper.codehaus.org`, 20130111.

[6] Retreieved from `http://www.jboss.org/drools/drools-fusion.html`, 20130111.

[7] Retrieved `http://esper.codehaus.org/tutorials/tutorial/tutorial.html` 20130110.

5.3 Stream Reasoning

The term ‘stream reasoning’ has emerged in the last few years. Initially, this has applied to the means to process streams of RDF triples, using SPARQL extended with concepts similar to those discussed above [10].From a technical point of view, there would appear to be very little difference between the two, since both rely upon finite state machines to recognise patterns in the input stream in order to trigger some action. The important advance offered by C-SPARQL, however, is that the event data may also contain reference to ontologies, and not just literals. As a result, data from different sources may be associated semantically, rather than through the syntactic structure of the stream records, thus reducing the coupling (in the software engineering sense) between producer and consumer.

Alternative, but different, logic-based approaches are put forward by Gebser et al [23] and Anicic-et-al [5]. The difference is that the former is based on answer set semantics, while that latter uses Prolog, but both have to face technical challenges to ensure that the tree synthesis approach both works efficiently and does not consume unbounded resources. The essential idea is that the head of a logic programming term denotes the recognition of a complex event, subject to the satisfaction of the right hand side of the clause. As such, the programmer is presented with a language that has much in common with real-time expert systems, but the implementations must provide temporal performance guarantees as well as a means to ‘forget’ facts in order to recover memory. As a result, both approaches incorporate a substantial amount of theoretical work that establishes the correctness of their mechanisms to recover memory and to compute results under real-time constraints.

6 Closing Remarks

We have put forward a personal survey of the situation in COIN and how its strengths and weaknesses fit with cloud computing. In particular, we believe that the resource-oriented architecture and stream processing have much to offer in conjunction with COIN technologies. In conclusion, we make some suggestions for next steps:

1. Make our components deployable as Software as a Service (SaaS), possibly through RESTful [39] interfaces and informed by a ROA perspective,
2. Out-source complex situation analysis to software built for the purpose, rather than trying to embed it in an agent architecture that was not designed for the task,
3. Collect datasets from cloud systems for testing and training of decision-making components, but not forgetting the value of synthetic datasets as a means to test boundary conditions,
4. Utilise communication protocols and data representations that enable interoperability, such as enterprise bus architectures [6,27,40] or distributed communication architectures [11] with semantically-annotated messaging for further decoupling of producer and consumer, and
5. Take advantage of free cloud services to prototype and share ideas and services.

Acknowledgements. I am grateful to the organizers of COIN 2012 at AAMAS for inviting me to speak at the meeting and to have the opportunity to follow that up with this invited paper. Some of ideas presented here have been developed in part with the support of The Royal Society (UK), the University of Otago and the COST action on Agreement Technologies.

References

1. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011) ISBN: 978-3-642-19344-6
2. Alechina, N., Dastani, M., Logan, B.: Programming norm-aware agents. In: van der Hoek, et al. (eds.) [25], pp. 1057–1064.
3. Alexander, C.: A Timeless Way of Building. Center for Environmental Structure. Oxford University Press Inc., USA (1980)
4. Andrighetto, G., Villatoro, D., Conte, R.: Norm internalization in artificial societies. AI Communications 23(4), 325–339 (2010)
5. Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., Studer, R.: A Rule-Based Language for Complex Event Processing and Reasoning. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 42–57. Springer, Heidelberg (2010)
6. Apache Camel, `http://camel.apache.org/` (retrieved January 04, 2013)
7. Artikis, A.: Dynamic protocols for open agent systems. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) AAMAS, vol. (1), pp. 97–104. IFAAMAS (2009)
8. Athakravi, D., Corapi, D., Russo, A., Vos, M.D., Padget, J.A., Satoh, K.: Handling change in normative specifications. In: van der Hoek, et al. (eds.) [25], pp. 1369–1370.
9. Balke, T., De Vos, M., Padget, J.: Normative Run-Time Reasoning for Institutionally-Situated BDI Agents. In: Cranefield, S., van Riemsdijk, M.B., Vázquez-Salceda, J., Noriega, P. (eds.) COIN 2011. LNCS, vol. 7254, pp. 129–148. Springer, Heidelberg (2012)
10. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying RDF streams with C-SPARQL. SIGMOD Record 39(1), 20–26 (2010)
11. Bernstein, D., Vij, D.: Using XMPP as a transport in intercloud protocols. In: 2010 the 2nd International Conference on Cloud Computing, CloudComp (2010)
12. Cardoso, H.L., Oliveira, E.C.: Institutional reality and norms: Specifying and monitoring agent organizations. Int. J. Cooperative Inf. Syst. 16(1), 67–95 (2007)
13. Cliffe, O., De Vos, M., Padget, J.: Modelling Normative Frameworks Using Answer Set Programing. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 548–553. Springer, Heidelberg (2009)
14. Criado, N., Argente, E., Botti, V.: A BDI architecture for normative decision making. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 1383–1384. International Foundation for Autonomous Agents and Multiagent Systems (2010)
15. Dignum, V., Padget, J.: Multiagent organizations. In: Weiss, G. (ed.) Multiagent Systems, 2nd edn. MIT Press (2012) (in press)
16. El-Akehal, E.E.D., Padget, J.A.: Pan-supplier stock control in a virtual warehouse. In: Berger, M., Burg, B., Nishiyama, S. (eds.) AAMAS (Industry Track), pp. 11–18. IFAAMAS (2008)
17. Elakehal, E.E., Padget, J.: Market intelligence and price adaptation. In: Proceedings of the 14th Annual International Conference on Electronic Commerce, ICEC 2012, pp. 9–16. ACM, New York (2012), `http://doi.acm.org/10.1145/2346536.2346538`

18. Ennis, R.L., Griesmer, J.H., Hong, S.J., Karnaugh, M., Kastner, J.K., Klein, D.A., Milliken, K.R., Schor, M.I., Van Woerkom, H.M.: A continuous real-time expert system for computer operations. IBM J. Res. Dev. 30(1), 14–28 (1986), `http://dx.doi.org/10.1147/rd.301.0014`
19. Esteva, M., Padget, J.: Auctions without Auctioneers: Distributed Auction Protocols. In: Moukas, A., Ygge, F., Sierra, C. (eds.) AMEC 1999. LNCS (LNAI), vol. 1788, pp. 220–238. Springer, Heidelberg (2000), `http://dx.doi.org/10.1007/10720026_12`
20. Ferguson, I.A.: Touring machines: Autonomous agents with attitudes. Computer 25(5), 51–55 (1992), `http://dx.doi.org/10.1109/2.144395`
21. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis. University of California, Irvine (2000)
22. Fornara, N., Viganò, F., Verdicchio, M., Colombetti, M.: Artificial institutions: a model of institutional reality for open multiagent systems. Artif. Intell. Law 16(1), 89–105 (2008)
23. Gebser, M., Grote, T., Kaminski, R., Obermeier, P., Sabuncu, O., Schaub, T.: Stream reasoning with answer set programming: Preliminary report. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) KR. AAAI Press (2012)
24. Gordon, G.: The development of the general purpose simulation system (gpss). In: Wexelblat, R.L. (ed.) History of Programming Languages I, pp. 403–426. ACM, New York (1981), `http://doi.acm.org/10.1145/800025.1198386`
25. van der Hoek, W., Padgham, L., Conitzer, V., Winikoff, M. (eds.): International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, vol. 3. IFAAMAS (2012)
26. Hübner, J.F., Sichman, J.S., Boissier, O.: A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In: Bittencourt, G., Ramalho, G.L. (eds.) SBIA 2002. LNCS (LNAI), vol. 2507, pp. 118–128. Springer, Heidelberg (2002), `http://dx.doi.org/10.1007/3-540-36127-8_12`
27. Ibsen, C., Anstey, J.: Camel in Action. Manning (2010) ISBN-13: 978-1935182368
28. Laffey, T.J., Cox, P.A., Schmidt, J.L., Kao, S.M., Read, J.Y.: Real-time knowledge-based systems. AI Mag. 9(1), 27–45 (1988), `http://dl.acm.org/citation.cfm?id=44132.44133`
29. Laird, J.E., Newell, A., Rosenbloom, P.S.: Soar: An architecture for general intelligence. Artificial Intelligence 33(1), 1–64 (1987), `http://www.sciencedirect.com/science/article/pii/0004370287900506`
30. Lazer, D., Friedman, A.: The dark side of the small world: how efficient information diffusion drives out diversity and lowers collective problem solving ability. Program on Networked Governance (PNG) Working paper 06-001, Harvard University (2006), `http://www.hks.harvard.edu/netgov/files/png_workingpaper_series/PNG06-001_WorkingPaper_LazerFriedman.pdf` (retrieved January 06, 2013)
31. Lee, J., Baines, V., Padget, J.: Decoupling Cognitive Agents and Virtual Environments. In: Dignum, F. (ed.) CAVE 2012. LNCS, vol. 7764, pp. 17–36. Springer, Heidelberg (2013)
32. McFall, M.E., Klahr, P.: Simulation with rules and objects. In: Proceedings of the 18th Conference on Winter Simulation, WSC 1986, pp. 470–473. ACM, New York (1986), `http://doi.acm.org/10.1145/318242.318479`
33. Müller, J.P.: The Design of Intelligent Agents: A Layered Approach. LNCS, vol. 1177. Springer, Heidelberg (1996)
34. Neville, B., Pitt, J.: PRESAGE: A Programming Environment for the Simulation of Agent Societies. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) ProMAS 2008. LNCS, vol. 5442, pp. 88–103. Springer, Heidelberg (2009)

35. Nieves, J.C., Padget, J., Vasconcelos, W., Staikopoulos, A., Cliffe, O., Dignum, F., Vázquez-Salceda, J., Clarke, S., Reed, C.: Coordination, organisation and model driven approaches for dynamic, flexible, robust software and services engineering. In: Schahram, D., Li, F. (eds.) Service Engineering, pp. 85–115. Springer (2011) ISBN: 978-3-7091-0414-9, `http://dx.doi.org/10.1007/978-3-7091-0415-6_4`
36. Noriega, P.: Fencing the Open Fields: Empirical Concerns on Electronic Institutions (Invited Paper). In: Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J. (eds.) ANIREM and OOOP 2005. LNCS (LNAI), vol. 3913, pp. 81–98. Springer, Heidelberg (2006)
37. Nygaard, K., Dahl, O.J.: The development of the simula languages. In: Wexelblat, R.L. (ed.) History of Programming Languages I, pp. 439–480. ACM, New York (1981), `http://doi.acm.org/10.1145/800025.1198392`
38. Overdick, H.: The Resource-Oriented Architecture. In: 2007 IEEE Congress on Services Services, pp. 340–347 (2007), `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4278816`
39. Pautasso, C., Zimmermann, O., Leymann, F.: RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. In: 17th International World Wide Web Conference (WWW 2008), Beijing, China, pp. 805–814 (April 2008), `http://www2008.org/`
40. Ranathunga, S., Cranefield, S.: Embedding BDI agents in business applications using enterprise integration patterns (extended abstract). In: Ito, Jonker, Gini, Shehory (eds.) Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013). IFAAMAS (to appear, 2013)
41. Ranathunga, S., Cranefield, S., Purvis, M.K.: Identifying events taking place in second life virtual environments. Applied Artificial Intelligence 26(1-2), 137–181 (2012)
42. Richardson, L., Ruby, S.: RESTful Web Services, 1st edn. O'Reilly Media, Inc. (May 2007)
43. Rodriguez, J.: On the design and construction of agent-mediated electronic institutions. IIIA Monographs 14 (2001)
44. Rodríguez, J.A., Noriega, P., Sierra, C., Padget, J.: FM96.5 A Java-based Electronic Auction House. In: Proceedings of 2nd Conference on Practical Applications of Intelligent Agents and MultiAgent Technology (PAAM 1997), London, UK, pp. 207–224 (April 1997) ISBN 0-9525554-6-8, `http://www.iiia.csic.es/Projects/fishmarket/PAAM97.ps.gz`
45. Simon, H.A.: Rational choice and the structure of the environment. Psychological Review 63(2), 129–138 (1956)
46. Vázquez-Salceda, J., Dignum, V., Dignum, F.: Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems 11(3), 307–360 (2005)
47. Web Services Architecture, `http://www.w3.org/TR/ws-arch/#relwwwrest`, `http://www.w3.org/TR/ws-arch/#relwwwrest` (retrieved August 08, 2011)
48. Winikoff, M., Cranefield, S.: On the testability of BDI agent systems. Information Science Discussion Papers Series 2008/03. University of Otago (2008), `http://hdl.handle.net/10523/1063` (retrieved January 04, 2013)

Monitoring Interaction in Organisations

Mehdi Dastani[1], Leendert van der Torre[2], and Neil Yorke-Smith[3,4]

[1] Utrecht University, The Netherlands
M.M.Dastani@uu.nl
[2] University of Luxembourg, Luxembourg
leon.vandertorre@uni.lu
[3] Olayan School of Business, American University of Beirut, Lebanon
[4] University of Cambridge, United Kingdom
nysmith@aub.edu.lb

Abstract. In an organisational setting, such as an online marketplace, the organisation monitors agent interactions, and enforces norms by means of sanctions. This paper provides an operational semantics for agent interactions within such a setting, distinguishing constitutive norms for monitoring and sanction rules for enforcement of norms. Our contribution emphasizes a more detailed exploration of the processes of monitoring commitments created through agent interactions and imposition of sanctions when commitments are violated. We consider both agent–agent and agent–environment interactions, focusing on operationalizing enforcement of commitment-based norms. We provide a generic way to develop operational semantics from specific definitions of norm behaviour. For an example set of norm behaviours, we sketch some formal properties that follow from our semantics, such as continuity, (non-)interference, and (non-)redundancy.

1 Introduction

In an open multi-agent environment, agents can coordinate their interactions by means of communication. Following earlier works concerned with multi-agent organisations, we define the semantics of agent interactions, including communication actions, in terms of social commitments [20]. Such commitments constitute institutional facts of an organisation, and it is within an organisational setting such as 2OPL [12] that agents interact. For our purposes, the organisation consists of two main processes: the monitoring process checks for compliance to norms while the enforcement process ensures imposition of sanctions.[1] Agent interactions affect the state of the institutional facts (i.e., commitments). The norms, represented as counts-as rules, define both constitutive as well as regulative norms. Finally, sanction rules are responsible for updating the organisation state as a consequence of detected violations.

Our focus, then, is on what has been called *agreement technologies*, i.e., how coordination is achieved between autonomous computational entities [2]. Following [12] one can specify norms to govern the interaction between agents. However, we believe that interaction by means of communication should respect a set of generic norms which are

[1] Other organisation attributes and processes, such as environmental interaction, or roles, entities, and the relationships between them, are orthogonal to our purpose in this paper.

inherent in communication actions. We follow the line of works that posit that communication actions create and operate on social commitments [20,11,15]. In our setting, an organisation will manage the commitments according to the utterances (messages) and actions of the agents, and regiment the compliance of agents with the commitments. Since we represent norms as counts-as rules, we specify the relations between communication actions and commitments explicitly via counts-as rules.

Our contribution emphasizes a more detailed exploration of the processes of monitoring and enforcement through an organisation's tracking of institutional state defined in terms of commitments, which are established and modified through agent communication acts. By adopting a simple set of such actions, and not focusing on the protocol or semantic concerns of a full Agent Communication Language, we are able develop a more technical account for a full operational semantics, and explore its properties.

A common methodology to define an operational semantics in such a paradigm specifies the typical or desired behaviour of the system. For example, explicit acceptance of commitments (i.e., must the creditor accept a commitment, or not?), fulfilment of a commitment's consequent before its antecedent (i.e., can a non-detached commitment be satisfied?), or synchronization details (e.g., if a consequent is fulfilled at the same moment its deadline occurs, is the commitment detached or expired?). Based on such choices, one defines norm behaviour, typically based on a finite state automaton; based on the automaton, one then defines an operational semantics.

Our methodology differs in that, given an automaton defined using rules, we develop a generic way to go from the automaton to the operational semantics. Thus, the semantics for any given description of norm behaviours follow automatically by means of a generic mechanism. Hence our generic methodology can be applied to different sets of counts-as rules, and further can admit constitutive norms that change dynamically.

We use the term 'organisation' and 'organisational setting' in this paper. Some authors draw a distinction between institutional concepts and organisational concepts, placing for example in the former, concepts such as brute and institutions facts, counts-as relations, and the constitutive nature of commitments, and placing in the latter concepts such as roles, organisational objectives, and terminologies or conventions. In such a view, to be explored would be the relationship between the organisation and the institutional reality created by the conventions we study.

The remainder of the paper is organized as follows. Sections 2 and 3 present our formal setting and operational semantics of the organisation from the coordination point of view. Section 4 illustrates on an insurance scenario. Section 5 studies properties of our approach and investigates the management of agent interaction and commitments. Section 6 places our work in context of the existing literature, Section 7 presents topics for future research, and Section 8 concludes the paper with a summary.

2 Normative Organisation

Our model of an exogenous organisation allows individual agents to interact with each other and with their shared environment. The function of an organisation, as we will use the term, is to monitor and regulate the interaction between agents, i.e., it observes the (inter)actions of agents, evaluates their consequences and imposes sanctions if needed.

The agents' (inter)actions are assumed to be observed/received by the organisation as events. The evaluation of agents' (inter)actions as well as their sanctioning are realized based on the norms and sanctions that specify the organisation. In this paper, we focus on commitments as a specific type of norms and study agents' interactions that influence the generation and state of commitments.

For example, consider an organisation such as a marketplace where agents can buy and sell goods from each other. In such an organisation, agent i offers agent j to make a payment $p_{(b,20)}$ (i.e., paying 20 euro for book b) within 5 days if agent j first sends the book $s_{(b,i)}$ within 2 days. We assume that the marketplace has the credit card information of the participating agents and that the agents can give a payment order to the organisation when they have received their goods. We also assume that the marketplace is allowed to withdraw money from a credit card without the order from the card holder only in designated cases, e.g., when the buyer of a good, who has asked the seller to send him the good, returns the good back to the seller, the organisation can then subtract the shipping cost paid by the seller from the credit card of the buyer agent.

The organisation can monitor the agents' interactions, determine the commitments of agents, and ensure that the agents fulfill their commitments—or otherwise take necessary measures such as putting the violating agent on a blacklist. The organisation, as an exogenous process, cannot *intervene* in the decision making of individual agents by either disallowing them to perform actions or forcing them to perform specific actions, i.e., agents are autonomous and decide their own actions.

2.1 Agent Interactions

We begin by identifying possible actions that agents can perform to interact with each other or with their shared environment. These include pure communication actions, as well as non-communicative actions that change the actual state of their environment. It is important to notice that it is not our purpose to define an Agent Communication Language (ACL) or a communications protocol [14,1]. Instead, we select a representative set of actions that influence the generation and state of commitments. The following six actions will prove sufficient for justifying the adequacy of our organisational model for managing and enforcing commitments. We will use variables $x,\ y$, etc. to range over the agent names $i,\ j$, etc.; propositional variables $p,\ q$, etc. to range over propositions $a,\ b$, etc.; and finally $d,\ d'$, etc. to range over deadlines $t_m,\ t_n$, etc. where $m, n \in \mathbb{N}$.

- *offer*(x, y, p, q, d_1, d_2) — x tells y that x will make q true in the environment by deadline d_2 if p becomes true in the environment by deadline d_1
- *tell*(x, y, p) — x tells y that p is true in the environment
- *cancel*(x, y, q) — x tells y that x will not make q true in the environment
- *release*(y, x, q) — y tells x that x need not make q true in the environment
- *failure*(x, y, p) — x tells y that p cannot be made true in the environment
- *do*(x, p) — x performs an action to make proposition p true in the environment

In our running example, agent i can *offer* agent j to *do* the payment $p_{(b,20)}$ before time t_5 if agent j sends him the book $s_{(b,i)}$ before time t_2. Agent i can offer this deal to agent j by performing the following action: *offer*$(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$.

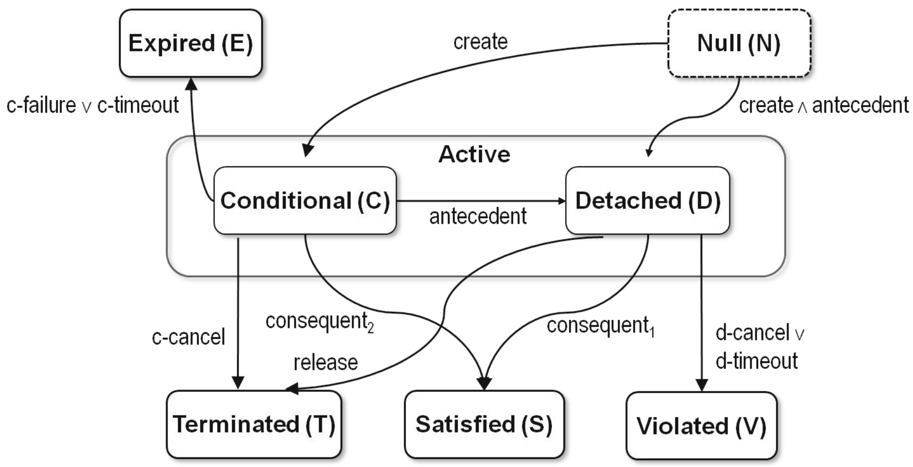


Fig. 1. State transitions of commitment lifecycle

2.2 Commitments

A *joint commitment* (hereafter *commitment*) is defined as a tuple $C = \mathsf{C}(x, y, p, q, d_1, d_2)$. Agent x (as debtor) tells agent y (as creditor) that if proposition p (the antecedent) is brought about by deadline d_1 then x will bring about q (the consequent) by deadline d_2. In the rest of the paper, we assume that deadline d_1 will be satisfied before deadline d_2. It is important to note that C is neither a second-order predicate nor a modal operator and that we do not aim at devising a logic to reason about the internal structure of commitments. $\mathsf{C}(x, y, p, q, d_1, d_2)$ is an atomic proposition denoting a specific commitment.

Fig. 1 shows the states of a lifecycle of a commitment, adapted from [21] (we omit suspension and delegation of commitments). Boxes indicate states and arrows indicate transitions. A commitment, once created, moves to state Conditional. Should the antecedent become true, the commitment moves to Detached. Should the consequent become true, it moves to Satisfied. However, should the antecedent not become true by d_1, then the commitment is Expired. Likewise if the consequent of a detached commitment does not become true by d_2, then the commitment is Violated. It is likewise Violated if x cancels a detached commitment. The commitment is Terminated if x cancels it before it is detached, or y releases x from C once it has been detached.

We will write commitment state with superscript, i.e., $\mathsf{C}^{\mathsf{state}}$. It will be useful to distinguish violation because of cancel (vc) and violated because of timeout (vt).

2.3 Organisation

An organisation is specified by facts, norms (including commitments), and sanctions. We distinguish *brute* and *institutional* facts. Brute facts denote the state of the shared environment (e.g., b_j denoting the fact that agent j has book b or $p_{(b,20)}$ denoting the fact that 20 euro is paid for book b), while institutional facts denote the normative state of an organisation (e.g., $\mathsf{C}^c(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$ denoting the fact that agent i is committed to pay 20 euro before t_5 if agent j sends book b before t_2). In the following, we assume Π_b and Π_i to be finite disjoint sets of brute and institutional facts (constructed by two disjoint sets of brute and institutional atomic propositions), respectively. Moreover, we follow [12] and represent norms by means of *counts-as* rules that relate brute and institutional facts. The original version of the counts-as construct is of the form "ϕ counts as ψ in the context c". We represent a counts-as construct as a rule $\phi \wedge c \Longrightarrow_{cr} \psi$, where $\phi, c \in \Pi_b$ and $\psi \in \Pi_i$. For example, an offer by agent i to agent j to do a payment if j sends i a book *counts-as* the creation of a conditional commitment. In the following, we use counts-as rules to evaluate and determine the institutional consequences of a certain environment state. In the more general framework proposed in [12], the antecedent of counts-as rules can include institutional facts as well. This allows institutional facts to be created based on other institutional facts making it possible to specify more complex norms such as country-to-duty norms. Finally, we represent sanctions by rules of the form $\phi \Longrightarrow_{sr} \psi$, where $\phi \in \Pi_b \cup \Pi_i$ and $\psi \in \Pi_b$. Note the difference between counts-as and sanctions rules: the first relates brute to institutional facts, while the latter relates institutional facts to brute facts. In our running example, a delay in payment by agent i beyond day 5 will be considered as a violation and will be sanctioned by having agent i on the organisation's blacklist.

We would like to emphasize that the consequent of the sanction rules (in this case having an agent on the blacklist) are brute facts and not actions. In our proposed model, an organisation imposes sanctions by updating its brute state with the consequents of the applicable sanction rules. It is also important to note that sanction rules are not meant to create other commitments, but they are meant to realize the final punishments. The creation of new commitments based on other commitments can in principle be modelled through counts-as rules that allow institutional facts in their antecedents.

Definition 1. *An* organisation *is specified as* (F, cr, sr)*, where* $\mathsf{F} \subseteq \Pi_b$ *is a set of initial brute facts,* cr *is a set of counts-as rules, and* sr *is a set of sanction rules.* □

In our running example, the organisation is initially specified by some brute facts such as agent j wants to sell a book b, and the blacklist of the organisation is empty, i.e., $b_j \in \mathsf{F}$. The set of facts will change during the execution of multi-agent system, based on the interaction between agents, e.g., the fact $p_{(b,20)}$ will be added to F when an agent pays 20 euro for book b. Note that the institutional facts such as commitments will be generated only during the execution of multi-agent system as a consequence of agents' interactions. Although an organisation can be specified in terms of arbitrary norms [12], we suppose that specific agents' interaction creates and manipulate social commitments (institutional facts) and claim that manipulation of commitments should respect a specific set of norms. Therefore, we focus on commitment-based norms and represent them as counts-as rules defined in terms of specific actions.

Fig. 2 illustrates a set of commitment-based norms, represented as counts-as rules. We next explain these rules, which specify how interactions between agents operate on social commitments. The application of counts-as rules (and the resulting removal and addition of commitments) are explained later in this section when we present the operational semantics of our organisational model in transition rules **1**–**4**, which are introduced in Section 3, below.

1 $\mathit{offer}(x,y,p,q,d_1,d_2) \Longrightarrow_{cr} \mathsf{C}^c(x,y,p,q,d_1,d_2)$
2 $\mathit{tell}(y,x,p) \wedge \mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge \neg d_1 \wedge p \Longrightarrow_{cr} \mathsf{C}^d(x,y,p,q,d_1,d_2)$
3 $\mathit{do}(y,p) \wedge \mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge \neg d_1 \wedge p \Longrightarrow_{cr} \mathsf{C}^d(x,y,p,q,d_1,d_2)$
4 $\mathit{tell}(x,y,q) \wedge \mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge \neg d_1 \wedge q \Longrightarrow_{cr} \mathsf{C}^s(x,y,p,q,d_1,d_2)$
5 $\mathit{do}(x,q) \wedge \mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge \neg d_1 \wedge q \Longrightarrow_{cr} \mathsf{C}^s(x,y,p,q,d_1,d_2)$
6 $\mathit{tell}(x,y,q) \wedge \mathsf{C}^d(x,y,p,q,d_1,d_2) \wedge \neg d_2 \wedge q \Longrightarrow_{cr} \mathsf{C}^s(x,y,p,q,d_1,d_2)$
7 $\mathit{do}(x,q) \wedge \mathsf{C}^d(x,y,p,q,d_1,d_2) \wedge \neg d_2 \wedge q \Longrightarrow_{cr} \mathsf{C}^s(x,y,p,q,d_1,d_2)$
8 $\mathit{cancel}(x,y,q) \wedge \mathsf{C}^d(x,y,p,q,d_1,d_2) \wedge \neg d_2 \wedge \neg q \Longrightarrow_{cr} \mathsf{C}^{vc}(x,y,p,q,d_1,d_2)$
9 $\mathsf{C}^d(x,y,p,q,d_1,d_2) \wedge d_2 \wedge \neg q \Longrightarrow_{cr} \mathsf{C}^{vt}(x,y,p,q,d_1,d_2)$
10 $\mathit{failure}(y,x,p) \wedge \mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge \neg d_1 \wedge \neg p \Longrightarrow_{cr} \mathsf{C}^e(x,y,p,q,d_1,d_2)$
11 $\mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge d_1 \wedge \neg p \Longrightarrow_{cr} \mathsf{C}^e(x,y,p,q,d_1,d_2)$
12 $\mathit{cancel}(x,y,q) \wedge \mathsf{C}^c(x,y,p,q,d_1,d_2) \wedge \neg d_1 \wedge \neg p \Longrightarrow_{cr} \mathsf{C}^t(x,y,p,q,d_1,d_2)$
13 $\mathit{release}(y,x,p) \wedge \mathsf{C}^d(x,y,p,q,d_1,d_2) \wedge \neg d_2 \wedge p \Longrightarrow_{cr} \mathsf{C}^t(x,y,p,q,d_1,d_2)$

Fig. 2. Counts-as rules specify the lifecycle of commitments based on actions and deadlines

- Performing action "x offers to y that x realizes q before d_2 if y realizes p before d_1" counts as creation of a conditional commitment (superscript c denotes conditional state of commitment; similar convention is used for other commitment states). The application of rule #1 by the organisation adds institutional fact $\mathsf{C}^c(x,y,p,q,d_1,d_2)$ to the institutional facts.
- Performing action "y tells x that p is realized" or "y does act and realizes p" when d_1 is still not passed counts as detaching the conditional commitment. The application of these rules #2 or #3 removes the conditional commitment from institutional facts and adds a corresponding detached commitment to it.[2]
- Performing action "x tells y that q is realized" or "x does act and realizes q" when d_1 is still not passed counts as satisfying the conditional commitment. The application of these rules #4 or #5 removes the conditional commitment from institutional facts and adds a corresponding satisfied commitment to it.
- Performing action "x tells y that q is realized" or "x does act and realizes q" when d_2 is still not passed counts as satisfying the detached commitment. The application of these rules #6 or #7 removes the detached commitment from institutional facts and adds a corresponding satisfied commitment to it.
- Performing action "x cancels to realizes q" when d_2 is still not passed counts as the violation of the detached commitment. The application of this rule #8 removes the detached commitment and adds a corresponding violated commitment.
- Elapsing deadline d_2 counts as the violation of a detached commitment. The application of this rule #9 removes the detached commitment from institutional facts and adds a corresponding violated commitment to it.
- Performing action "y fails to realize p" when d_1 is still not passed counts as expiration of the conditional commitment. The application of this rule #10 removes the conditional commitment and adds a corresponding expired commitment.

[2] Detaching a commitment based on telling assumes that y is a trusted agent, i.e., its utterances are according to its beliefs. Note that an organisation may develop a list of trusted agents.

- Elapsing deadline d_1 counts as expiration of a conditional commitment. The application of this rule #11 removes the conditional commitment from institutional facts and adds a corresponding expired commitment to it.
- Performing action "x cancels to realize q" when d_1 is still not passed counts as termination of the conditional commitment. The application of this rule #12 removes the conditional commitment and adds a corresponding terminated commitment to it.
- Performing action "y releases a detached commitment after p has been satisfied" when d_2 is still not passed counts as termination of the conditional commitment. The application of this rule #13 removes the conditional commitment from institutional facts and adds a corresponding terminated commitment to it.

Finally, organisations can be specified in terms of arbitrary sanctions represented by *sanction rules* [7]. Sanctions are defined in terms of specific violations and determine how a violated system state can be turned back to a 'normal' state by means of a system update. In contrast to commitments, sanctions are not generic and depend on application in hand. For example, in our marketplace organisation, a timed-out commitment caused by agent i who has failed to make a payment will be sanctioned by blacklisting the agent. This sanction can be represented as:

$$\mathsf{C}^{vt}(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5) \Longrightarrow_{sr} \mathit{blacklist}_i$$

Another possible sanction can be designed to cope with a detached commitment that is cancelled by its debtor. For example, suppose agent j sends the book before deadline t_2 after which agent i cancels the commitment. The *cancel* action by i violates the detached commitment $\mathsf{C}^d(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$, turning it into the canceled commitment $\mathsf{C}^{vc}(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$. One may want to sanction such a violation by charging i the shipping cost paid by j (and possibly some additional administration costs). Such a sanction can be represented by the following rule, which indicates that agent i should pay the shipping cost ($\mathit{chargedShippingCost}_{(i,5)}$):

$$\mathsf{C}^{vc}(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5) \Longrightarrow_{sr} \mathit{chargedShippingCost}_{(i,5)}$$

3 Operational Semantics

We now give the operational semantics for the normative organisation. The specification of an organisation determines the its initial state. The execution of an organisation is determined by a set of transition rules that specify possible transition steps. In the following, we first define the (initial) states of an organisation, followed by the set of four transition rules.

Definition 2. *The* state of an organisation *is specified as the tuple $\langle \sigma_b, \sigma_i, cr, sr \rangle$, where $\sigma_b \subseteq \Pi_b$, $\sigma_i \subseteq \Pi_i$, cr is the set of counts-as rules, and sr is the set of sanction rules. Since the counts-as and sanction rules do not change during the execution of organisations, we omit cr and sr from the organisation states, representing state as $\langle \sigma_b, \sigma_i \rangle$. Let (F, cr, sr) be the initial specification of an organisation. The* initial state *of the organisation is $\langle \mathsf{F}, \emptyset \rangle$, i.e., the initial set of institutional facts is the empty set.* □

In order to compute the set of commitments (and sanctions) that should be generated or modified by the counts-as rules cr (and the sanction rules sr) in a given state of an organisation, we follow [7] and define the *closure* of a set of propositions under a set of rules. Let $\mathsf{cond}(()r)$ and $\mathsf{cons}(()r)$ denote the condition and consequent of rule r, respectively. First, we determine the rules R that are applicable to a set of propositions X as:

$$\mathsf{App}(X, R) = \{r \in R \mid X \models \bigwedge \mathsf{cond}(r)\}$$

Using function $\mathsf{cl}_X^R(Y) = \{\mathsf{cons}(r) \mid r \in \mathsf{App}(X \cup Y, R)\}$ we define $\mathsf{cl}_X^R{\uparrow}^\omega$ as the smallest fixed-point of $\mathsf{cl}_X^R(\cdot)$, which exists due to Knaster/Tarski's fixed point theorem [7]. This fixed point provides the set of heads of all applicable rules.

Let cr be the set of counts-as rules presented in Fig. 2 and $\mathsf{cl}_X^{cr}{\uparrow}^\omega$ be the closure of set X under cr. The following transition rule specifies the interaction between two agents through communication action α. Note that the organisation updates the institutional facts (e.g., commitments) based on the performed communication action and by applying the above counts-as rules.

$$\frac{com(\alpha) \ \& \ \sigma_i' = \oplus(\sigma_i \cup \sigma_b \cup \{\alpha\})}{\langle \sigma_b, \sigma_i \rangle \rightarrow_c \langle \sigma_b, \sigma_i' \rangle} \tag{1}$$

1. $com(\alpha)$ indicates that α is the communicated message,
2. $\oplus(X) \ = \sigma_i \setminus \{c^x(\boldsymbol{V}) \mid c^y(\boldsymbol{V}) \in (\mathsf{cl}_X^{cr}{\uparrow}^\omega \setminus X)\} \cup \ (\mathsf{cl}_X^{cr}{\uparrow}^\omega \setminus X)$,
   for $x, y \in \{c, d, s, vc, vt, e, t\}$.

According to this transition rule, the state of a multi-agent organisation can make a transition when message α is communicated. In this transition the institutional state σ_i of the organisation is updated by applying the counts-as rules to determine new institutional facts. The new institutional facts are computed by taking the closure of institutional facts, brute facts, and the performed action under the counts-as rules, i.e., $\mathsf{cl}_X^{cr}{\uparrow}^\omega \setminus X$. Note that the original set of institutional facts, brute facts, and the performed action denoted by X are *removed* from the closure to obtain the new institutional facts. The second item in the rule specifies the update of the institutional facts by adding new institutional facts to σ_i while removing the corresponding institutional facts from σ_i. This operation guarantees that the state of commitments ($x, y \in \{c, d, s, vc, vt, e, t\}$) changes according to the transitions in the commitment lifecycle depicted in Fig. 1. Lastly, note that $\mathsf{cl}_X^{cr}{\uparrow}^\omega \setminus X = \emptyset$ if none of the counts-as rules are applicable. In our running example, the performance of action *offer*$(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$ in an organisation state $\langle \sigma_b, \sigma_i \rangle$ causes a transition to state $\langle \sigma_b, \sigma_i \cup \{\mathsf{C}^c(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)\}\rangle$ where a commitment is created. This is accomplished by the application of rule 1.

The next transition rule specifies the performance of *non-communicative* actions, such as an agent sending a book. Note that counts-as rules 3 and 5 cover the cases where performing a non-communicative action by some agent counts as changing the state of a commitment.

$$\frac{act(\alpha) \ \& \ \sigma_b' = \otimes(\sigma_b, \alpha) \ \& \ \sigma_i' = \oplus(\sigma_i \cup \sigma_b' \cup \{\alpha\})}{\langle \sigma_b, \sigma_i \rangle \rightarrow_a \langle \sigma_b', \sigma_i' \rangle} \tag{2}$$

1. $act(\alpha)$ indicates that α is the action,
2. $\otimes$ is an update operation that changes the state of environment σ_b after performing α. We assume the existence of such an update operator.
3. $\oplus$ is the operator for updating institutional facts as defined above.

The new institutional facts σ_i' are determined based on σ_b', the result of realizing the effect of α on σ_b. In our running example, sending the book b by agent j (i.e., we have that $\alpha = s_{(b,i)}$) causes a transition of the organisation state from $\langle \sigma_b, \sigma_i \rangle$, where we have $\sigma_i \models \mathsf{C}^c(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$, to $\langle \sigma_b', \sigma_i' \rangle$ with $\sigma_i' \not\models \mathsf{C}^c(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$ and $\sigma_i' \models \mathsf{C}^d(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$. Of course, we assume this is only possible if $\sigma_b \models s_{(b,i)} \wedge \neg t_2$ indicating that the sending action is indeed performed within the corresponding deadline t_2. This is accomplished by the application of counts-as rule 3.

Finally, we allow the environment state to change by the internal mechanism of the environment, e.g., the state of a clock changes automatically. This is essential for the application of counts-as rules 7 and 9, which are applicable when the deadline elapses.

$$\frac{\sigma_b \rightarrow \sigma_b' \;\&\; \sigma_i' = \oplus(\sigma_i \cup \sigma_b')}{\langle \sigma_b, \sigma_i \rangle \rightarrow_e \langle \sigma_b', \sigma_i' \rangle} \tag{3}$$

This transition rule ensures that elapse of the deadlines are possible and changes the state of the institutional facts accordingly. Let the organisation of our running example be in state $\langle \sigma_b, \sigma_i \rangle$ where $\sigma_i \models \mathsf{C}^d(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$, $\sigma_b \not\models p_{(p,20)}$, and the environment makes a transition such that $\sigma_b' \models d_5$. In this case, we have $\sigma_i' \not\models \mathsf{C}^d(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$ and $\sigma_i' \models \mathsf{C}^{vt}(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$. This is accomplished by the application of the counts-as rule 7.

Sanctions. A common scenario is that the organisation imposes sanctions when commitments are violated. For example, a non-paying agent is added to a blacklist. We represent sanctions by rules connecting commitment violations to a specific brute state of the organisation. In order to allow context-dependent sanctions, the antecedent of rules can be composed of both brute and institutional facts. Given a set of sanction rules sr and the state of an organisation $\langle \sigma_b, \sigma_i \rangle$, we define the enforcement of sanctions as the closure of state under the sanction rules, denoted by $\mathsf{cl}^{sr}_{(\sigma_b \cup \sigma_i)} {\uparrow^\omega} \setminus \sigma_i$. The following transition rule ensures that the organisation imposes sanctions when commitments are violated.

$$\frac{\sigma_b' = \mathsf{cl}^{sr}_{(\sigma_b \cup \sigma_i)} {\uparrow^\omega} \setminus \sigma_i \;\;\&\;\; \sigma_i' = \sigma_i \setminus \mathit{sanctioned}(\sigma_i, \sigma_b', sr)}{\langle \sigma_b, \sigma_i \rangle \rightarrow_s \langle \sigma_b', \sigma_i' \rangle} \tag{4}$$

The function $\mathit{sanctioned}(\sigma_i, \sigma_b', sr)$ removes all violated commitments from σ_i for which sanctions are imposed. Consider again the sanction for the payment violation in our running example, which is represented by the rule

$$\mathsf{C}^{vt}(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5) \Longrightarrow_{sr} blacklist_i$$

and let the organisation be in the state $\langle \sigma_b, \sigma_i \rangle$ where $\sigma_i \models \mathsf{C}^{vt}(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$. The application of this transition rule causes a transition to the state $\langle \sigma_b', \sigma_i' \rangle$ where $\sigma_b' \models blacklist_i$ (i.e., i is blacklisted) and $\sigma_i' \not\models \mathsf{C}^{vt}(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$.

Multi-agent System Execution. Recall that an organisation has two main processes to perform: (1) monitoring the interaction between agents, and between agents and the environment, to detect norm violations; and (2) enforcing norms by means of sanctions when violations are detected. In our framework, transitions **1**–**3** are responsible for the monitoring task while transition **4** is responsible for imposing sanctions. The executions of a multi-agent organisation are determined by the transition system which is specified by the transition rules **1**–**4**; the system consists of all possible computational runs.

Definition 3. *Given transition rules **1**–**4** and an initial state of an organisation* $c_0 = \langle \sigma_b, \sigma_i \rangle$, *a* computational run $CR(c_0)$ *is an infinite sequence* $c_0, c_1, \ldots$ *where* c_i *is a multi-agent organisation state and* $\forall_{i>0} : c_i \rightarrow c_{i+1}$ *is a transition derived by applying transition rules **1**–**4**. The* execution *of a multi-agent organisation with initial state* c_0 *is a set of all possible computational runs* $CR(c_0)$. □

Possible computational runs of an execution are due to different orders of the applications of transition rules **1**–**4**. One may constrain the set of possible computational runs by specifying one specific order for applying transition rules. For example, one order may apply transition rules **1**–**4** consecutively while a second order applies transition rules **1**–**3** interspersed by the application of transition rule **4**. According to the first order, the organisation allows (1) agents to communicate, (2) agents to interact with the environment, (3) the environment to make a transition, and (4) the organisation to impose sanctions. By contrast, the second order imposes sanctions immediately after each of the three activities. Note that the first order allows different violations to occur before sanctions are imposed while the second order applies the sanctions immediately. The consequences of these different orders will be studied in the next section.

4 Example

Chopra et al. [9] describe an insurance claim process involving a vehicle repair, with the actors being the driver claimant (assumed to be not at fault, and whose policy has no deductible), the driver's insurance company, a car repair garage, and a damage assessor. In our example scenario, the agents are a Customer (who owns the car), an Insurer (who pays for repairs), a Repairer (who conducts the repair), and an Assessor (who decides how much the repair should cost). Commitments between agents model the business protocols of the process:[3]

- C_1: insurer to repairer: if insurance has been validated and the repair has been reported, then the insurer will have paid and approved the assessment within 7 days
- C_2: insurer to assessor: if the assessment has been done, the assessment will have been paid within 5 days
- C_3: assessor to repairer: if damages have been reported and the insurance has been validated, a damage assessment will have been performed within 2 days
- C_4: repairer to customer: if the insurance has been validated and the car was damaged, then the car will have been repaired within 8 days

[3] Note there is an 'implied' commitment from insurer to customer: if the insurance has been validated and the car was damaged, then the car will have been repaired.

- C_5: insurer to customer: if the premium has been paid, then the insurance will have been validated within 8 days

Hence, formally, we have a set of social commitments as follows. Note that we specify relative deadlines for the consequent of commitments using the notation $+x$ to indicate x days after the antecedent becomes true. $\emptyset$ indicates an absence of a deadline.

- $C_1 = \mathsf{C}(\text{Insurer, Repairer, insurance-validated} \wedge \text{repair-reported, assessment-approved}$ $\wedge\text{payment-done}, \emptyset, +7)$
- $C_2 = \mathsf{C}(\text{Insurer, Assessor, assessment-done, assessment-paid}, \emptyset, +5)$
- $C_3 = \mathsf{C}(\text{Assessor, Repairer, damages-reported} \wedge \text{insurance-validated, assessment-done}, \emptyset, +2)$
- $C_4 = \mathsf{C}(\text{Repairer, Customer, insurance-validated} \wedge \text{car-damaged, car-repaired}, \emptyset, +8)$
- $C_5 = \mathsf{C}(\text{Insurer, Customer, premium-paid, insurance-validated}, \emptyset, +8)$

In the scenario, the car owner reports an accident to her insurance company, and takes the car to a garage. On certification that the insurance is valid (because the customer has paid the insurance premium), the repair garage accepts the damaged car and contacts the assessor. Since the insurance is valid, the repair garage commences the repair. The assessor reports to the insurance company, which approves the assessment of damage and pays the assessor for its work. The repair garage reports to the insurance company when it has completed the repair, and with the approval of the insurer, the garage then tells the customer that the car is ready. The insurer pays the repair garage.

Consider now a set of agent actions corresponding to this scenario. These actions are manifest in communication actions, and the impact of the actions upon the commitment store. The trace in Table 1 begins with the *offer*s that create the five commitments. Column Rule gives the Counts-as Rule. We abbreviate the logical propositions by their initials, e.g., pp for premium-paid. In line 6, the customer proves payment of the insurance. In line 9, she reports car damage. In line 14, the garage has completed the repair. In line 17, the garage tells the customer the car is ready.

5 Properties

The propositions in this section show various kind of properties following from our approach and operational semantics. We describe them but leave the proofs and further formalization for future work.

5.1 Temporal Properties

We study first *temporal properties*, i.e., the dynamics of the commitment states. These properties illustrate that the commitment states follow the commitment lifecycle in Fig. 1. In this subsection, we assume that each *offer*(x, y, p, q, d_1, d_2) is done with unique p and q, such that there is no interference. We discuss the lifting of this assumption in the following subsection.

We begin by stating continuity properties about conditional and detached commitments: they hold until they are fulfilled or the deadline is reached. We use Linear Time Logic LTL with the *release* operator, not to be confused with the release communication action of Section 2.1: ϕ releases ψ if ψ is true until the first position in which ϕ is true (or forever if such a position does not exist). We assume execution traces to be fair.

Table 1. Execution trace in car insurance scenario

Step	Agent	Communication Act	Time	Rule	Institutional State	Brute Facts
0	–	–	0	–	$\{\}$	$\{\}$
1	I	*offer*(I, R, iv ∧ rr, aa ∧ pd)	1	1	$\{C_1^C\}$	$\{\}$
2	I	*offer*(I, A, ad, ap)	2	1	$\{C_1^C, C_2^C\}$	$\{\}$
3	A	*offer*(A, R, dr ∧ iv, ad)	3	1	$\{C_1^C, C_2^C, C_3^C\}$	$\{\}$
4	R	*offer*(R, C, iv, cr)	4	1	$\{C_1^C, C_2^C, C_3^C, C_4^C\}$	$\{\}$
5	I	*offer*(I, C, pp, iv)	5	1	$\{C_1^C, C_2^C, C_3^C, C_4^C, C_5^C\}$	$\{\}$
6	C	*do*(C, pp)	6	3	$\{C_1^C, C_2^C, C_3^C, C_4^C, C_5^D\}$	$\{pp\}$
7	C	*do*(I, iv)	7	5	$\{C_1^C, C_2^C, C_3^C, C_4^C, C_5^S\}$	$\{iv, pp\}$
8	–	–	8	–	$\{C_1^C, C_2^C, C_3^C, C_4^C, C_5^S\}$	$\{cd, iv, pp\}$
9	C	*tell*(C, R, iv ∧ cd)	9	2	$\{C_1^C, C_2^C, C_3^C, C_4^D, C_5^S\}$	$\{cd, iv, pp\}$
10	R	*tell*(R, A, dr ∧ iv)	10	2	$\{C_1^C, C_2^C, C_3^D, C_4^D, C_5^S\}$	$\{cd, dr, iv, pp\}$
11	A	*do*(A, ad)	12	5	$\{C_1^C, C_2^C, C_3^S, C_4^D, C_5^S\}$	$\{ad, cd, dr, iv, pp\}$
12	A	*tell*(A, I, ad)	12	2	$\{C_1^C, C_2^D, C_3^S, C_4^D, C_5^S\}$	$\{ad, cd, dr, iv, pp\}$
13	I	*tell*(I, A, ap)	16	4	$\{C_1^C, C_2^S, C_3^S, C_4^D, C_5^S\}$	$\{ad, ap, cd, dr, iv, pp\}$
14	R	*do*(R, rr)	16	3*	$\{C_1^C, C_2^S, C_3^S, C_4^D, C_5^S\}$	$\{ad, ap, cd, dr, iv, pp, rr\}$
15	R	*tell*(R, I, iv)	17	2*	$\{C_1^D, C_2^S, C_3^S, C_4^D, C_5^S\}$	$\{ad, ap, cd, dr, iv, pp, rr\}$
16	I	*tell*(I, R, aa)	17	4*	$\{C_1^D, C_2^S, C_3^S, C_4^D, C_5^S\}$	$\{aa, ad, ap, cd, dr, iv, pp, rr\}$
17	R	*tell*(R, C, cr)	18	2	$\{C_1^D, C_2^S, C_3^S, C_4^S, C_5^S\}$	$\{aa, ad, ap, cd, cr, dr, iv, pp, rr\}$
18	I	*do*(I, pd)	21	5*	$\{C_1^S, C_2^S, C_3^S, C_4^S, C_5^S\}$	$\{aa, ad, ap, cd, dr, iv, pd, pp, rr\}$

Proposition 1. *If the organisation updates the institutional facts based on the performed communication action as follows:*

- offer(x, y, p, q, d_1, d_2)*, then* $p \vee q \vee d_1$ *releases* $\mathsf{C}^c(x, y, p, q, d_1, d_2)$.
- *if* tell(y, x, p)*, and* $\mathsf{C}^c(x, y, p, q, d_1, d_2)$ *holds, then* $\neg d_1 \wedge p \dashrightarrow (q \vee d_2)$ *releases* $\mathsf{C}^d(x, y, p, q, d_1, d_2)$
- *if we have* do(y, p)*, and* $\mathsf{C}^c(x, y, p, q, d_1, d_2) \wedge \neg d_1 \wedge p$ *holds, then* $q \vee d_2$ *releases* $\mathsf{C}^d(x, y, p, q, d_1, d_2)$

Second, we can state continuity properties about reaching termination states: each conditional and detached commitment state will lead to precisely one termination state.

Proposition 2. *For any commitment, we have at most one of* $\mathsf{C}^c(x, y, p, q, d_1, d_2)$*,* $\mathsf{C}^d(x, y, p, q, d_1, d_2)$*,* $\mathsf{C}^s(x, y, p, q, d_1, d_2)$*,* $\mathsf{C}^{vc}(x, y, p, q, d_1, d_2)$*,* $\mathsf{C}^{vt}(x, y, p, q, d_1, d_2)$*,* $\mathsf{C}^e(x, y, p, q, d_1, d_2)$*, and* $\mathsf{C}^t(x, y, p, q, d_1, d_2)$ *true at any time.* □

Third, we can state continuity properties about some of the termination states: once they are true, they will stay true. This does not hold for the violation states, as the violation will be removed once the sanction is applied.

Proposition 3. *If* $\mathsf{C}^s(x, y, p, q, d_1, d_2)$*,* $\mathsf{C}^e(x, y, p, q, d_1, d_2)$*, or* $\mathsf{C}^t(x, y, p, q, d_1, d_2)$ *hold, then they will hold forever.* □

Further temporal properties can be defined to illustrate that the operational semantics behaves according to the commitment lifecycle in Fig. 1. Since the characterization of these properties is relatively straightforward, we turn here to two other classes of properties which can be defined for our operational semantics.

5.2 Interference

Figure 1 is a bit misleading in the sense that in practice there is not a single commitment cycle at the same time, but many of them operate in parallel. In this section we consider whether one commitment cycle can affect another one; we call this interference. We now consider the lifting of the assumption that each *offer*(x, y, p, q, d_1, d_2) is done with unique p and q. For example, consider the situation where an agent i makes two consecutive *offer* statements, *offer*$(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$ and *offer*$(i, j, t_{(b,i)}, p_{(b,20)}, t_2, t_5)$.

Proposition 4. *If the organisation updates the institutional facts based on a performed communication action* tell(y, x, p) *or an action* do(y, p)*, then it can detach or satisfy multiple commitments at once.* □

Example 1. Assume the organisation updates the institutional facts based on the following actions: *offer*$(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$, *offer*$(i, j, t_{(b,i)}, p_{(b,20)}, t_2, t_5)$, *do*$(j, s_{(b,i)})$, and *do*$(j, t_{(b,i)})$. This leads to two detached commitments, namely $\mathsf{C}^d(i, j, s_{(b,i)}, p_{(b,20)}, t_2, t_5)$ and $\mathsf{C}^d(i, j, t_{(b,i)}, p_{(b,20)}, t_2, t_5)$. Performing *do*$(i, p_{(b,20)})$ by agent i will now move both detached commitments to satisfied commitments, i.e., agent i de-commits itself by paying 20 euro instead of 40 euro.[4] □

[4] Another similar example is when an agent makes two offers that differ only in the deadlines. In that case, the commitment with the earlier deadline 'subsumes' the one with the later deadline.

This example illustrates that the agents must be careful to distinguish propositions. In this case, the agents should syntactically distinguish the two propositions referring to pay 20 euro, and it is left to the agent to prevent this undesired interference.

5.3 Redundancy

Redundancy properties concern communication actions that, if omitted as part of a sequence of communication actions, do not change the final set of institutional facts [3]. We confine ourselves to an example of a redundancy property, as follows.

Consider a sanction system where agents are blacklisted regardless of the commitment they violated, i.e., containing the following two rules:

$$\mathsf{C}^{vt}(x, y, p, q, d_1, d_2) \Longrightarrow_{sr} add_{y,BL}$$
$$\mathsf{C}^{vc}(x, y, p, q, d_1, d_2) \Longrightarrow_{sr} add_{y,BL}$$

In the context of our running example, we say that an offer is *blacklist redundant* if it does not change the situations in which an agent will be blacklisted. Given a sequence of offers, which offers can be made in addition which are in this sense blacklist redundant?

Proposition 5. – *If* $\mathsf{C}^c(x, y, p, q, d_1, d_2)$*, then* offer$(x, y, p \wedge r, q, d_1, d_2)$ *is blacklist redundant. If I have offered you that if you do* p *I will do* q*, then it is redundant to offer you that if you do* $p \wedge r$ *then I will do* q*, under the same conditions.*

– *If* $\mathsf{C}^c(x, y, p, q{\wedge}r, d_1, d_2)$*, then* offer$(x, y, p, q, d_1, d_2)$ *is blacklist redundant. If I have offered you that if you do* p *I will do* $q \wedge r$*, then it is redundant to offer you that I will do* q*, under the same conditions.*

– *If* $\mathsf{C}^c(x, y, p, q, d_1, d_2){\wedge}\mathsf{C}^c(x, y, r, q, d_1, d_2)$*, then* offer$(x, y, p{\vee}r, q, d_1, d_2)$ *is blacklist redundant. If I have offered you that if you do* p *I will do* q*, and if you do* r *I will do* q*, then it is redundant to offer you that if you do* $p \vee r$ *then I will do* q*, under the same conditions.*

– *If* $\mathsf{C}^c(x, y, p, q, d_1, d_2){\wedge}\mathsf{C}^c(x, y, p, r, d_1, d_2)$*, then* offer$(x, y, p, q{\wedge}r, d_1, d_2)$ *is blacklist redundant. If I have offered you that if you do* p *I will do* q*, and if you do* p *I will do* r*, then it is redundant to offer you that I will do* $q \wedge r$*, under the same conditions.*

The redundancy properties give some deeper insight in the system, but they depend on the sanctioning system adopted by the organisation. The above properties still hold when the sanction system does not depend on the actual commitment that is violated, but only that *some* commitment has been violated. Another way of reasoning says that the sanction depends only on q—not on p, the agents x and y, or the deadlines. This yields a subset of the above properties. A further refinement is to assume relations among sanctions. For example, a sanction for violating the offer to do p should not be more than that for violating the offer to do $p \wedge q$. We leave this matter for future work.

6 Related Work and Discussion

The idea of non-communicative actions and the enforcement of sanctions by monitoring the state of commitments is something common in the agent literature. The schema of our model is standard when dealing with commitments in agents organizations: organizations consist of facts, norms, commitments and sanction, the agents can perform a set of actions which lead to the creation of commitments, and these commitments have rules in form of count-as rules and when the commitments are violated, sanctions apply [20,11,17,14,15,19,8,16,4,1,9,18,10].

Moreover, formalizations are often based on a lifecycle for commitments using an operational semantics. However, the aim is typically to give a semantics for a large variety of speech acts, for example for propose or request, and they therefore have to define their semantics in considerable detail. Further still, the trend is to make the languages ever more complex, for example by introducing higher order commitments, meta-commitments [22], embedded temporal regulations [1], goal of organizations, other norms than commitment-based ones, and so on.

Building on earlier work [11,14], Fornara et al. [15] propose an ACL based on communication actions. They define norms as “rules that manipulate commitments of the agents engaged in an interaction” and thus, unlike our work, define norms as event-driven rules and provide a more limited operational semantics. Our work is further distinguished in generality since we can define a range of operational semantics by changing the constitutive norms. Our purpose is not to define an ACL but to establish how norms are be operationalized in an organisation setting.

Compared to earlier works, our approach differs in the following regards. First, we use counts-as rules explicitly as technical constructs while Fornara et al. [14], for instance, treat counts-as relation primarily as linguistic conventions. Note that counts-as rules in our work can be seen as defeasible rules. Second, we provide an operational semantics for interactions (among which communication actions) within an organisational setting. Fornara et al. provide semi-formal specification of organisations and consider only communication actions. Further, we analyze the properties of interaction within an organisational setting. Third, we consider the effect of non-communicative actions as well as the elapse of deadlines. Fourth we have sanction rules while earlier works leave open the question of what should happens when commitments are violated. Fifth, in contrast to some works, we adopt a contemporary lifecycle of commitments, following Chopra and Singh [10] and precedents.

These last authors are interested in simplifying the specification of commitment-based protocols or requirements. Our aim is in line with the latter paper [10], and the two approaches are orthogonal. Chopra and Singh capture business requirements with commitment-based specifications, and then group these specifications into reusable methods. They consider protocol enactment but not the organisational setting, which is our focus. By contrast, we do not aim to directly model business requirements, but start from the lifecycle of commitments and define a minimal language to make all possible changes to the commitment base.

An alternative approach, that aims for the same level of genericity as we pursue, uses deontic-inspired specifications that are monitored and enforced, such as the recent work of Álvarez-Napagao and colleagues [13]. An interesting discussion for the future is to compare our commitment-based approach versus a deontic-inspired approach.

7 Future Work

Our on-going work is to characterise the properties of the operational semantics and to prove them. In this paper, we considered two principal communication actions: *offer* and *tell*. Alternative semantics can be defined for these actions using other constitutive rules, and compared to the ones we defined. Moreover, our model can be extended to a wider range of communication actions and provide their semantics in terms of social commitments. Extending the set of communication actions should be done carefully since commitments may create unwanted interferences.

Second, commitments, as we have adopted them, possess special slots for deadlines. Alternatively, as proposed by Marengo et al. [18], the deadline could be part of the propositional content (the antecedent and consequent) of the commitment. In that case, the deadline would not have a special status. For example, we could express the commitment that the insurer will have paid the assessment within 7 days, and (further) approved it within 14 days. Whether such composite commitments can be expressed by several commitments in our language, and in general the advantages and disadvantages of the two definitions from the point of view of the organisational setting, is left for further research.

Third, we considered here generic counts-as rules that were designed based on the semantics of specific communication actions (i.e., *offer* and *tell*). A possible extension would be to investigate the interaction between such generic counts-as rules and domain-specific counts-as rules representing domain-specific norms. For example, we can consider norms such as “buying a book while having no money on a credit card counts-as a violation”.

Fourth, a further issue not considered in this paper is the role of interaction protocols in agents’ interactions. The representation of an organisation could be enriched with—in addition to counts-as and sanction rules—interaction protocols that constrain the order of communication actions.

Finally, the dynamics of organisations is an important topic which we have not treated here. We think there is potential in exploring a mechanism to dynamically change the rules of the constitutive norms and sanctions [6,5].

8 Summary

This papers focuses on the use of social commitments as institutional facts. The lifecycle of these commitments can be managed by an organisation based on agents’ interactions, and thus the commitments play a role in coordination of agents and their interaction, through the monitoring and enforcement processes of the organisation.

The state of an organisation consists of brute and institutional facts, a set of counts-as rules, and a set of sanction rules. In addition, we define the closure of a set of propositions under a set of rules. Then we introduce an operational semantics using four transition rules. The first transition rule defines the interaction between two agents through communication actions, the second specifies the performance of non-communicative actions, the third defines the change by the internal mechanism of the environment, such as the state of the clock, and the fourth ensures that the organisation imposes sanctions when commitments are violated. We define a multiagent system execution as a sequence of states of the organisation, where each transition is derived by applying one of the four transition rules.

We define a normative organisation using six actions of the agents: conditional offers to make something true (*offer*), telling that something is true (*tell*), performing an action to make something true (*do*), cancelling one's own offer to make something true (*cancel*), cancelling someone else's offer to make something true (*release*), and informing that something cannot be made true (*failure*). In addition, we define seven states for commitments: Null, Conditional, Detached, Expired, Terminated, Satisfied, and Violated. We introduce thirteen counts-as rules specifying the lifecycle of commitments based on communication actions, non-communicative actions, and deadline expirations. In addition, we give two examples of sanction rules.

We illustrate the operational semantics and the normative organisation using an insurance claim process. We define five commitments, from insurer to repairer, from insurer to assessor, from assessor to repairer, from repairer to customer, and from insurer to customer. An execution trace of seventeen actions and one deadline expiration illustrates how these commitments change over time.

We discuss *temporal properties*, *interference*, and *redundancy*. First, the temporal properties are the usual specification and verification properties defined for operational semantics. Second, the interference properties refer to the possible interaction between offers. For example, can a single communication act detach multiple commitments? Third, the redundancy properties, which are more involved than temporal and interference properties, give a form of implicit logical relations between the communication actions. We consider a sanction system where agents are blacklisted regardless of the commitment they violated, and we show, for example, that weaker commitments and some conjunctions of commitments are blacklist redundant.

Our approach presents a generic methodology that can be applied to different sets of counts-as rules: for instance, variants of Fig. 2 that reflect variants of the commitment lifecycle, e.g., if conditional commitments are created only after explicit acceptance by the creditor. It further can permit constitutive norms that change dynamically.

Acknowledgements. We thank the referees for their constructive comments, and the participants of the COIN'12 workshop at AAMAS'12 for the discussions.

References

1. Baldoni, M., Baroglio, C., Marengo, E.: Behavior-oriented commitment-based protocols. In: Proc. ECAI, pp. 137–142 (2010)
2. Billhardt, H., Centeno, R., Cuesta, C.E., Fernández, A., Hermoso, R., Ortiz, R., Ossowski, S., Pérez-Sotelo, J.S., Vasirani, M.: Organisational structures in next-generation distributed systems: Towards a technology of agreement. Multiagent and Grid Systems 7(2-3), 109–125 (2011)
3. Boella, G., Broersen, J., van der Torre, L.: Reasoning about Constitutive Norms, Counts-As Conditionals, Institutions, Deadlines and Violations. In: Bui, T.D., Ho, T.V., Ha, Q.T. (eds.) PRIMA 2008. LNCS (LNAI), vol. 5357, pp. 86–97. Springer, Heidelberg (2008)
4. Boella, G., Damiano, R., Hulstijn, J., van der Torre, L.: A common ontology of agent communication languages: Modelling mental attitudes and social commitments using roles. Applied Ontology 2(3-4), 217–265 (2007)

5. Boella, G., Pigozzi, G., van der Torre, L.: Normative framework for normative system change. In: Proc. AAMAS, pp. 169–176 (2009)
6. Boella, G., van der Torre, L.: Regulative and constitutive norms in normative multiagent systems. In: Proc. KR, pp. 255–266 (2004)
7. Bulling, N., Dastani, M.: Verification and implementation of normative behaviours in multi-agent systems. In: Proc. IJCAI 2011, pp. 103–108 (2011)
8. Carabelea, C., Boissier, O.: Coordinating agents in organizations using social commitments. Electronic Notes in Theor. Comp. Sci. 150(3), 73–91 (2006)
9. Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 113–128. Springer, Heidelberg (2010)
10. Chopra, A.K., Singh, M.P.: Specifying and applying commitment-based business patterns. In: Proc. AAMAS, pp. 475–482 (2011)
11. Colombetti, M.: A commitment-based approach to agent speech acts and conversations. In: Proc. Workshop on Agent Languages and Communication Policies, pp. 21–29 (2000)
12. Dastani, M., Grossi, D., Meyer, J.-J.C., Tinnemeier, N.: Normative Multi-agent Programs and Their Logics. In: Meyer, J.-J.C., Broersen, J. (eds.) KRAMAS 2008. LNCS, vol. 5605, pp. 16–31. Springer, Heidelberg (2009)
13. Felipe, L.O., Álvarez-Napagao, S., Vázquez-Salceda, J.: Towards a framework for the analysis of regulative norm performance in complex networks. In: Proc. of 1st Intl. Conf. on Agreement Technogolies (AT 2012), pp. 103–104 (October 2012)
14. Fornara, N., Colombetti, M.: A commitment-based approach to agent communication. Applied Artificial Intelligence 18(9-10), 853–866 (2004)
15. Fornara, N., Viganò, F., Colombetti, M.: Agent communication and artificial institutions. Autonomous Agents and Multi-Agent Systems 14, 121–142 (2007)
16. Kibble, R.: Speech acts, commitment and multi-agent communication. Computational and Mathematical Organization Theory 12, 127–145 (2006)
17. Mallya, A.U., Yolum, P., Singh, M.P.: Resolving Commitments among Autonomous Agents. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, pp. 166–182. Springer, Heidelberg (2004)
18. Marengo, E., Baldoni, M., Baroglio, C., Chopra, A.K., Patti, V., Singh, M.P.: Commitments with regulations: Reasoning about safety and control in REGULA. In: Proc. AAMAS, pp. 467–474 (2011)
19. Pasquier, P., Chaib-draa, B.: Integrating Social Commitment-Based Communication in Cognitive Agent Modeling. In: Dignum, F.P.M., van Eijk, R.M., Flores, R. (eds.) AC 2005. LNCS (LNAI), vol. 3859, pp. 76–92. Springer, Heidelberg (2006)
20. Singh, M.P.: A Social Semantics for Agent Communication Languages. In: Dignum, F., Greaves, M. (eds.) Issues in Agent Communication. LNCS, vol. 1916, pp. 31–45. Springer, Heidelberg (2000)
21. Telang, P.R., Singh, M.P.: Specifying and verifying cross-organizational business models. IEEE Trans. Services Computing 4 (2011)
22. Venkatraman, M., Singh, M.P.: Verifying compliance with commitment protocols. Autonomous Agents and Multi-Agent Systems 2(3), 217–236 (1999)

Reasoning over Norm Compliance via Planning

Sofia Panagiotidi[1], Javier Vázquez-Salceda[1], and Frank Dignum[2]

[1] Universitat Politècnica de Catalunya, Spain
{panagiotidi,jvazquez@lsi.upc.edu}
[2] Utrecht University, The Netherlands
dignum@cs.uu.nl

Abstract. Norms are a way to provide some flexibility to the specification of acceptable actor behaviour in a shared context. Instead of viewing norms as static restrictions over an agent's conduct (and autonomy), the full power of normative specifications comes when norms are seen as guidelines that agents can use in their decision-making. In literature there is a lot of work on norm theories, models and specifications on how agents might take norms into account when reasoning, but many of them focus on the goal or intention selection and few of them apply the norms into the agent's plan generation. In this paper we present a norm-oriented agent that takes into consideration operationalised norms during the plan generation phase, using them as guidelines to decide the agent's future action path. In our work norms can be obligations or prohibitions which can be violated, and are accompanied by repair norms in case they are breached. To make norms influence plan generation, our norm operational semantics is expressed as an extension/on top of STRIPS semantics, acting as a form of temporal restrictions over the trajectories (plans) computed by the planner. In combination with the agent's utility functions over the actions, the norm-aware planner computes the most profitable trajectory concluding to a state of the world where the effects of all the active norms have been explored, including the repair norms. We use a simplified fire emergency scenario in order to demonstrate the usefulness of the framework, integrating the norm-aware planner to 2APL agent architecture. We depict possible outcomes depending on criteria such as time and danger.

Keywords: normative systems, decision making, reasoning, planning.

1 Introduction

In the last decade, computational systems have become more and more complex resulting in highly complicated interconnected networks. Agent-oriented models, techniques and methodologies provide some ways to tackle the complexity in distributed (software) systems. But as systems grow to include hundreds of components, the agent metaphor alone is not enough, and the need to design social or organisational autonomous models on top where the members' behaviour is somehow regulated in order to produce desirable and avoid undesirable situations becomes stronger.

A traditional way of dealing with possible deviations from desired behaviour is applying hard constraints to the agents. Still, this might lead to exponentially large numbers of decision paths and most of the times such a hard coded modelling leads to rigid and complex agent specification. For example, while it is possible for a firefighter agent to allow a restriction "whenever there is a fire, call reinforcements before entering the building", it might not always be the desired thing to do and under some specific circumstances (if for example the nearest fire station is far, implying that the reinforcements might take too long to arrive) one might desire to deviate from such a restricting measure. On the other hand, modern approaches ensuring appropriate individual entities' behaviour in distribute systems, which comes from multi-agent systems research, use norms (or regulations or policies) and/or communication protocols to express a different layer of desired or undesired states. From the individual's perspective, an agent needs to be able to function in an environment where norms act as behavioural restrictions or guidelines over what is appropriate, not only for the individual but also for the community.

While the very notion of agent autonomy refers to the ability of individual agents to determine their own actions, plans and beliefs and, to the capability of an agent to act independently, exhibiting control over its own internal state, it is not always clear how norms affect an agent's autonomy. In this paper we try to tackle and provide a practical solution to the following question:

"*How to model an agent that is able to take the environment's normative influence into account in its decision-making?*"

Given the above, we take our previous work [13] a step further and suggest that decision making in a normative environment can be implemented via a extended planning mechanism. The mechanism, integrated into an agents deliberation cycle, produces and at the same time evaluates the plans. In possession of a set of norms which can be seen as indications over desired behaviour of the agent and an objective, and taking into account the current state of affairs, it computes the most beneficial way to achieve its objective, that is, what sequence of actions should be followed in order to reach it gaining at the same time optimal cost/benefit though fulfilling or ignoring obligations and prohibitions.

The rest of the paper is structured as follows. In Section 2 we detail a motivating example which later will be used in order to provide implementational details. In Section 3 we describe the proposed architecture of an agent and the agent lifecycle and the tools used in order to integrate the normative planner into the agent platform. A formalisation of the planning and normative model is provided in Sections 4) and 5). The implementation details of the normative agent and the implementation of the scenario and the simulation outcomes are explained Section 7.

2 Example

The example follows a simple scenario where a building is on fire and the firefighters are called to help. The main firefighter arrives at the entrance. The environment (see the small graphics window inside Figure 5) consists of six doors in two corridors, the rooms behind which are on fire. In order to be able to extinguish a fire in a room, the door to that room needs to be open. The topographic knowledge is known to the agent, i.e. which door is next to which and where the entrance is located with reference to the doors. The agent can move freely from the entrance to the doors and back. The goal of the agent is to try and extinguish all the fires behind the doors while keeping the risk of getting injured during the operation as low as possible. The urgency of such realistic scenario lies on the fact that there could be victims that need to be saved behind the doors, still, for simplicity reasons we avoid modelling them.

One norm is put into force under these circumstances. Whenever the agent is about to enter a building on fire, he must be accompanied by reinforcements. More specifically, because the opening of a door produces a substantial amount of danger, the agent should make sure to be accompanied before any doors are opened. In the opposite case, a second norm is put into force, that is, his ranking position gets lowered (the ranking represents some rewarding bonus expressing the braveness of the firefighters and thus making it desirable to maintain it high).

We introduce three factors that make the example realistic and at the same time serve as parameters that the planner will take into account.

- **time** increases with every action that the agent performs. Additionally, whenever reinforcements are called, it always takes them some time to arrive to the spot.
- The **personal risk** (danger) of the agent increases with every action that the agent performs since time is passing by and the risk of getting hurt by a constantly increasing fire is bigger. Whenever the agent opens a door, his personal risk increases, still when he is accompanied by reinforcements this increase is less than when he is alone.
- The agent starts with an initial **ranking** level.

The objective is to have an agent operating in such an environment and additionally being able to come up with plans over how to fulfil the goal while at the same time keeping the combined value of the three factors as low as possible.

In order to create a more realistic simulation, we designed the environment to include some factors that the agent is unaware of. This results in different outcomes for similar scenarios. One such factor is that the agent always runs a random risk of dying when opening a door while being alone. This change is relatively small, but it still leads to cases where the simulation ends without the expected outcome. Furthermore, the environment might unexpectedly notify the agent about the danger of being inside the building getting too severe. In this case, the agent will need to make sure to take this information under consideration, dropping the plan and the goals and replanning taking the new information into account.

3 Architecture

As we mentioned in section 1, unlike most frameworks, our approach takes into consideration the norms during the planning phase, this is what we call "norm-oriented planning". To do this, we introduce in the agent's deliberation cycle (more concretely in its practical reasoning step) a norm-aware planning component that can create plans influenced by the norms.

We have chosen **2APL** [2] as the basis for our norm-aware agents. 2APL is a modular BDI-based agent programming language. At the individual agent level, 2APL agents are implemented in terms of beliefs, goals, actions, plans, events, and three types of rules (to generate plans for achieving goals, to process -internal and external- events and received messages, to handle and repair failed plans). The beliefs and goals of 2APL agents are implemented in a declarative way, while plans and (interfaces to) external environments are implemented in an imperative programming style. 2APL agents can perform different types of actions such as belief update actions, belief and goal test actions, external actions (including sense actions), actions to manage the dynamics of goals, and communication actions. Plans consist of actions that are composed by a conditional choice operator, iteration operator, sequence operator, or non-interleaving operator. It should be noted that a 2APL agent can observe an environment either actively by means of a sense action or passively by means of events generated by the environment.

The execution of an individual 2APL agent program is realised by a cyclic sense-reason-act process (deliberation process) including five steps: 1) Rules are checked and any new plans produced are added to the plan list. 2) Execution of the first action of each plan. After these actions, goals are queried in the belief base. Valid goals in the base are considered achieved and are removed from the goal base and any plan triggered by this goal is removed. 3) Processing of external events. 4) Handling of failed plans. 5) Processing of external messages

In our architecture (Figure 1) we have modified the way the deliberation process handles plans. While standard 2APL uses pre-computed plans, in our architecture 2APL requests plans dynamically to a planner. We have chosen **Metric-FF**, a domain independent planning system, as the basis for our normative planner. Metric-FF can deal with problems expressed in PDDL 2.1 level 2, combined with ADL [6]. Metric-FF can handle effects and constraints on linear functions over numerical state variables and favours minimisation of a given cost function. It uses Enforced hill-climbing (EHC), a heuristic search strategy, as the base search method. The algorithm combines Hill-climbing with systematic breadth first search. Unlike standard Hill-climbing it is enforced that the successor picked at the current state at each stage of the search is one that is better than the current state. Enforced Hill-Climbing solves the problem of local maxima by switching to breadth-first search if it gets trapped in a local maximum.

For the Metric-FF planner to be able to use the environment norms and the agent's beliefs and goals, we need to properly include them into the planner's inputs. The next sections explain the details of the norm-oriented planning

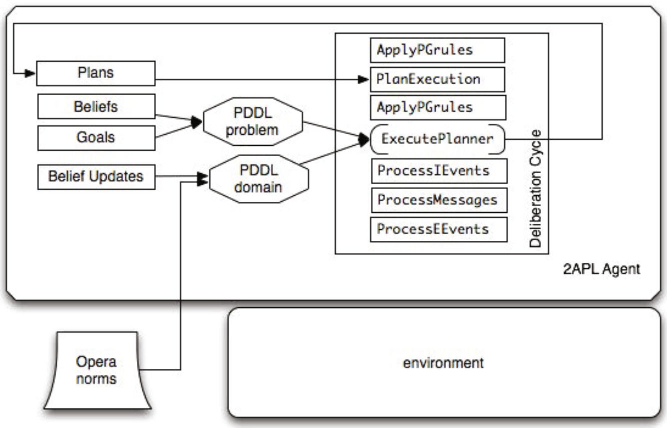


Fig. 1. 2APL Planner Architecture

semantics and how this is used in real-time scenarios. In sections 4 and 5 we adapt our norm semantics to the planner, and in section 6 we translate also the agent beliefs and goals into PDDL.

4 STRIPS Planning

In our work we use a situation calculus formalisation of STRIPS. Furthermore, we extend STRIPS with additional normative elements (see Section 5) in order to allow for normative reasoning within the planning process. In this section, we briefly describe the semantics of STRIPS [4].

Definition 1. *We define **F** as set of **fluents** $F = \{f_1, f_2, \ldots, f_m\}$ where fluent f_i is an atomic proposition (propositional property).*

*The state of the world is defined in terms of fluents that hold at the particular situation. We define a **state** to be a (possibly empty) subset of F, i.e. a state is represented by the set of fluents that are true in it. The fluents not in the state are assumed false.*

Each combination of fluents forms a different state, and the union of all the states is a set of states as in Definition 2.

Definition 2. *We define **S** to be the **state domain** (set of all states) occurring from F as $S = 2^f$.*

Definition 3. *Having a set of fluents F as in Definition 1, we define **A** to be a set of domain **actions** (actions with pre and postconditions) $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ where $\alpha_i =< C_{prec_i}, C_{post_i} >$ and C_{prec_i}, C_{post_i} are sets of fluents or negated fluents from F. Since states are represented by sets of conditions, the **transition function** $\longmapsto$ relative to a domain instance $< F, A >$ is a function $\longmapsto: S \times A \rightarrow$*

S where S is the state domain S occurring from F. If σ_1, $\sigma_2 \in S$ and $\alpha \in A$ and $\longmapsto (\sigma_1, \alpha) = \sigma_2$ then we write $\sigma_1 \overset{\alpha}{\longmapsto} \sigma_2$. The transition function can be defined as follows, using the simplifying assumption that actions can always be executed but have no effect if their preconditions are not met:

- *$\longmapsto (\sigma, < C_{prec_i}, C_{post_i} >)$ is $\sigma \cup \{$all m where m belongs to $C_{post_i}\}\backslash\{$all fluents l where $\neg l$ belongs to $C_{post_i}\}$ if $\{$every fluent j in C_{prec_i} belongs in $\sigma\}$ and $\{$every fluent k where $\neg k$ in C_{prec_i} does not belong in $\sigma\}$*
- *σ otherwise*

The function $\longmapsto$ can be extended to sequences of actions by the following recursive equations:

$$\sigma \overset{[]}{\longmapsto} = \sigma$$
$$\sigma \overset{[\alpha_1,\alpha_2,\dots,\alpha_n]}{\longmapsto} = \sigma \overset{\alpha_1}{\longmapsto} \overset{[\alpha_2,\dots,\alpha_n]}{\longmapsto}$$

Definition 4. *A **goal** g is a pair $< P, R >$, where P and R specify which fluents are true and false, respectively, in order for a state to be considered a goal state.*

In order to proceed to the definition of a plan, we assume the existence of an initial state σ_0 where $\sigma_0 \in S$. Then:

Definition 5. *A **plan** is a sequence of actions such that the state that results from executing the actions in order from the initial state satisfies the goal conditions. Formally, $[\alpha_1, \alpha_2, \dots, \alpha_n]$ is a plan for goal $g =< P, R >$ if the state $\sigma' =\longmapsto (\sigma_0, [\alpha_2, \dots, \alpha_n]))$ is such that $P \subseteq \sigma'$ and $R \cap \sigma' = \emptyset$.*

The above means that the state reached by the execution of the actions $\alpha_1, \dots, \alpha_n$ should lead to a state which satisfies the specification of the goal.

5 Normative Model

As explained in Section 1, our motivating force is how to deal with decision making and planning within normative environments. How can we represent domain knowledge and the normative influence so that the agent simulates a "natural" planning process? When might an agent decide to violate a norm? Could it predict this based on the available actions and preferences?

Using the domain elements of the STRIPS semantics (fluents, actions, states, goals, plans) we include norms as the additional elements that will lead to the definition of our normative model. In brief, the norms are expressed on top of STRIPS, acting as a complicated form of restrictions over the trajectories (plans) computed by the planner. In combination with utility functions over the actions, the system computes the most profitable trajectory concluding to a state of the world where no norms awaiting settlement exist.

5.1 Norms

To ease operationalisation, our norm representation is a tuple containing a deontic statement and the conditions for norm activation, deactivation and violation. The *deontic operator* of the deontic statement expresses the deontic "flavour" of the norm (in our case obligation or prohibition) and thereby establishes whether an agent should attempt to fulfil the norm or on the contrary avoid the prohibition. The *activation*, *maintenance* and *deactivating conditions* are specified as (partial) state descriptions, denoting the conditions that express when the norm gets activated, violated and deactivated. They add operational information to the norm, to simplify the verification and enforcement of the norm. They work as follows[1]:

- The *activation condition* specifies when a norm becomes active, i.e the state of affairs in which the norm is triggered (and must henceforth be checked for completion/violation);
- The *deactivating condition* specifies when the norm has been deactivated, i.e. has no longer normative force.
- The *maintenance condition* is needed for checking violations of the norm; it expresses the state of affairs that should hold all the time between the activation and the deactivation of the norm.

In essence, when a norm has been activated, has not yet expired and the maintenance condition is not fulfilled, a *violation* of the norm happens. Whenever a norm is violated, the norm continues to be active but in addition a norm violation triggers the activation of a new norm (a *repair norm*). This happens by including the "violation" of the original norm in the activating condition of the repair norm. Finally, a norm might have several active instances at the same time [11]. Below we define the set of norms N:

Definition 6. *Given a set of fluents F as in Definition 1 we define N to be a set of **norms** $N = \{N_1, N_2, \ldots, N_v\}$ where $N_i = \{id, type, C_{act}, C_{deact}, C_{maint}\}$ and id is an identificator, type is in $\{obligation, prohibition\}$ and $C_{act}, C_{deact}, C_{maint}$ are sets of fluents or negated fluents from F.*

5.2 Definition of Normative Model

We define our normative model below.

Definition 7. *A **normative model** is a tuple $M = (F, A, N)$ where F is a set of fluents as defined in Definition 1, A is the set of domain actions (labels) as in Definition 3 and N a set of norms as defined in Definition 6.*

In order to be able to know whether a norm is active or violated, it is not enough to be aware of the current state of affairs. An additional knowledge concerning the norm status at previous states is required. For example, in order to derive

[1] The operational semantics follow our previous work on norm lifecycle semantics [11].

that a norm gets deactivated one needs to know not only that the deactivation condition holds at a specific state but also that the norm was active previously. Therefore, such properties need to be defined with respect to an entire sequence of states (trajectory path) visited during the execution of a plan, starting from an initial point.

Definition 8. *Given a set of actions A, a plan $\pi = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ and an initial state σ_0, π generates the trajectory $< \sigma_0, \sigma_1, \ldots, \sigma_n >$ iff for every $1 \leq i \leq n$: $\sigma_i \xmapsto{\alpha_i} \sigma_{i+1}$*

We define F_{ext} as $F_{ext} = F \cup \{act(N), deact(N), maint(N), active(N), violated(N)\}$ where $act, deact, maint$ are special additional fluents indicating the activating, deactivating, maintenance and violation deactivation conditions of a norm N and $active, violated$ are fluents indicating the state of the norm (whether it is active or not and whether it is violated). As explained, the entailment for the norm lifecycle properties is defined with respect to a trajectory path produced by a plan in Definition 9.

Definition 9. *The interpretation of the entailment for the norm lifecycle in M over a trajectory $< \sigma_0, \sigma_1, \ldots, \sigma_n >$ of a path π at a state σ given a norm $N = \{id, type, C_{act}, C_{deact}, C_{maint}\}$ is as follows:*

- $\Big((M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models act(N)\Big) \Leftrightarrow \Big(\forall f \in C_{act} \cdot f \in \sigma_i \text{ and } \forall \neg f \in C_{act} \cdot f \notin \sigma_i\Big) \text{ and } \Big(\forall N' \in M : violated(N') \in C_{act} \Rightarrow (M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models violated(N')\Big)$
- $\Big((M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models deact(N)\Big) \Leftrightarrow \Big(\forall f \in C_{deact} \cdot f \in \sigma_i \text{ and } \forall \neg f \in C_{deact} \cdot f \notin \sigma_i\Big)$
- $\Big((M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models maint(N)\Big) \Leftrightarrow \Big(\forall f \in C_{maint} \cdot f \in \sigma_i \text{ and } \forall \neg f \in C_{maint} \cdot f \notin \sigma_i\Big)$
- $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models active(N) \Leftrightarrow \Big((M, \sigma_i) \models act(N) \text{ and } \neg deact(N)\Big) \text{ or } \Big((M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_{i-1}) \models active(N) \text{ and } (M, \sigma_i) \models \neg deact(N)\Big)$
- *For N of type obligation:* $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models violated(N) \Leftrightarrow \Big((M, \sigma_i) \models \neg maint(N) \text{ and } (M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models active(N)\Big)$
- *For N of type prohibition:* $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models violated(N) \Leftrightarrow \Big((M, \sigma_i) \models maint(N) \text{ and } (M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_i) \models active(N)\Big)$

The first three ones declare when a norm's activating condition, deactivating condition and maintenance condition will hold. To evaluate the truthfulness of its activating condition one takes into account whether it is also a repair norm for some other norm that is violated (if it includes the "violated(N')" predicate for some norm N'). The fourth declares that a norm is active at a state if its activating condition holds at the current state and the deactivating condition does not, or if it was already active at the previous state of the trajectory and its deactivating condition does not hold at the current state. Note here that the activeness status of a norm is calculated recursively over the path trajectory, since it depends on whether it was active or not at the previous state. The fifth declares that a violation of an obligation occurs at some state if it is active, and its maintenance condition does not hold at the current state. The sixth declares respectively that a violation of a prohibition occurs at some state if it is active, and its maintenance condition holds at the current state.

5.3 The Normative Planning Problem

Having stated the above, we are finally able to define the exact nature of what normative planning is. In a given planning domain where additional norms acquire committing (obligation) or preventing (prohibition) force through paths and when the agent failing to comply with them needs to see to it that a repairing state is reached, the problem is to find such a plan that the final state achieves the goal, that there are no pending obligations open and that for all possible violations of obligations and prohibitions that might have occurred, the repair state has been accomplished.

Formally, given an initial state σ_0, a goal g and a normative model $M = (F, A, N)$ we are looking for a plan $\pi = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ generating the trajectory $< \sigma_0, \sigma_1, \ldots, \sigma_n >$ where all goals are accomplished and additionally the obligations that rise from other violated norms are not active:

- $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_n) \models g$
- Forall obligations $O \in N$: $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_n) \not\models violated(O) \Rightarrow$
  $(M, < \sigma_0, \sigma_1, \ldots, \sigma_n >, \sigma_n) \not\models active(O)$

In later PDDL versions, actions can have an associated cost function which might be combined with conditional effects. This allows for paths (plans) to have a total cost. We introduce action costs in our framework in order to be able to evaluate not only the norm conformance but also to calculate most beneficial paths while taking into account compliance to or violation and reparation of norms. When defining what the norm repair (penalty) state is, we require that any plan-solution that violates a norm also takes the necessary steps to achieve its repair state. By giving the appropriate cost function to the actions, the planner is able to determine the choice between a path that complies to a norm and a path that violates it and afterwards compensates by reaching the repair state. In this way, such complex reasoning over conformance to the soft restraints imposed by norms becomes part of the planning problem rather that an external issue.

6 Implementation of the Normative Planning in PDDL

In this section we detail the implemented simulation of the example in Section 2. Many implementation details are omitted due to lack of space but also in order to preserve the clarity of the message of the paper.

6.1 Intermediate States

The norm status lifecycle is not trivial to integrate in the planning process. The reason for this is that it is not enough to check at every state whether a norm is violated or not. It is also necessary to check whether the norm applies or still applies (is active) subsequently implying that the violation should be checked too.

For each norm, we introduce an additional intermediate state which the planner is forced to include between all actions for every produced plan, which comes

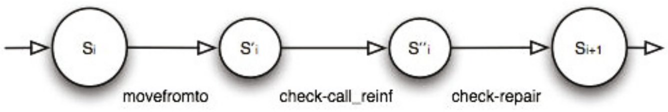


Fig. 2. Intermediate States

after state S_i and before state S_{i+1} (see Figure 2). That is so that we are able to implement the update of all norm statuses (activation, violation and activation of the repair norm), which, as seen in Section 5 are dependent on the previous state of affairs as well as the current one. This technique is known when implementing PDDL planners. In this way, the calculation of the norms lifecycle happens in two stages. In the first, the effects of the action are evaluated while norm status remains the same as it was before taking into consideration the action's effects. In the second step the norm status is evaluated. Figure 2 depicts the intermediate states for two norms introduced in the normative planner.

Figure 3 depicts the derivation rules for the norm lifecycle.

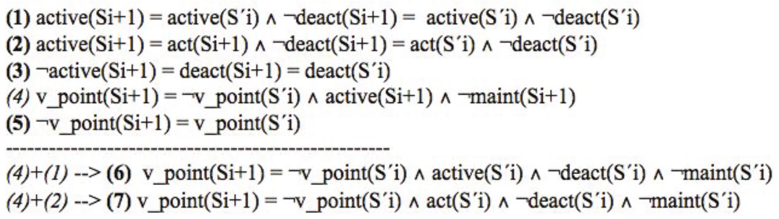

(1) active(Si+1) = active(S′i) ∧ ¬deact(Si+1) = active(S′i) ∧ ¬deact(S′i)
(2) active(Si+1) = act(Si+1) ∧ ¬deact(Si+1) = act(S′i) ∧ ¬deact(S′i)
(3) ¬active(Si+1) = deact(Si+1) = deact(S′i)
(4) v_point(Si+1) = ¬v_point(S′i) ∧ active(Si+1) ∧ ¬maint(Si+1)
(5) ¬v_point(Si+1) = v_point(S′i)
----
(4)+(1) --> **(6)** v_point(Si+1) = ¬v_point(S′i) ∧ active(S′i) ∧ ¬deact(S′i) ∧ ¬maint(S′i)
(4)+(2) --> **(7)** v_point(Si+1) = ¬v_point(S′i) ∧ act(S′i) ∧ ¬deact(S′i) ∧ ¬maint(S′i)

Fig. 3. Derivation rules

The rules follow an intuitive derivation pattern (the norm parameter N is omitted due to lack of space). The bold non italic are the ones that are expressed in terms of the previous state S'_i (and consequently included in the PDDL implementation).

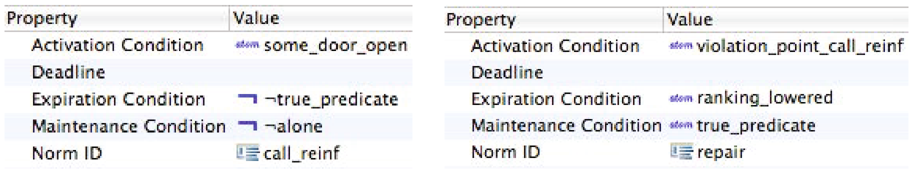
Since for every norm we introduce one intermediate state that evaluates its status (see Figure 2), every action in the plan is followed by $|N|$ states (with $|N|$ being the total number of the norms in the normative model). If $|V|$ is the vertex and $|E|$ is the edge number, with the addition of the intermediate states we will have $|V| + |N|$ and $|E| + |N|$ vertexes and edges in the search graph, augmenting the computational complexity respectively.

6.2 Cost Calculation Issues

By default the agent sets as criterion for the "preciousness" of a plan the minimisation of the addition of all the numerical factors. A more complex formula comprising of weights of importance for every factor might be defined within the agent and passed on to the normative planner.

The costs of a plan can be calculated in different ways. One can take the costs of repair actions completely into account, but also reduce them when for example the chance that someone discovers a violation is 0.5 and thus half the times repair actions do not have to be executed. This of course depends on the agents own morals as the detection chance might be of less interest to some agents. In addition, one can see the exercise of creating a plan and choosing the one with lowest cost as just one way of choosing the next action. This does not mean that the plan will always be executed completely. What happens in this case is to choose a next action BECAUSE it is the first one of a possible plan with the lowest cost to achieve a goal. But in principle the plan could be recomputed after each action, before a next action is chosen. This makes a difference. For example, an agent might go into the building, extinguish one fire and then (possibly under new data) replan and come outside the building and call for reinforcement. While if he originally had a plan to extinguish all fires right away he would never call reinforcement any more. Of course the overhead for replanning would have to be taken into account in the process but this is out of the scope of the paper.

6.3 Norms and Normative Planner Inputs



Property	Value
Activation Condition	some_door_open
Deadline	
Expiration Condition	¬true_predicate
Maintenance Condition	¬alone
Norm ID	call_reinf

(a) Prohibition: It is not permitted to be alone inside a burning building if any door is opened

Property	Value
Activation Condition	violation_point_call_reinf
Deadline	
Expiration Condition	ranking_lowered
Maintenance Condition	true_predicate
Norm ID	repair

(b) Obligation: In case of violation of the norm "call_reinf" the agent's ranking should be lowered

Fig. 4. Norms of the example

The norm (the agent is obliged to be accompanied by reinforcements when stepping into a burning building) is modelled as in Figure 4a. Note that the norm does not explicitly express the need for reinforcements, but instead forbids the agent to be alone in the building while any door is opened. This is done because the norms are expressed in terms of states rather than in terms of actions that are imposed or forbidden. The repair of the norm, a second norm stating that in the case of violation of the first the agent's ranking should be reduced, can be seen in Figure 4b.

The PDDL code for the intermediate step (as explained in Section 6) checking the status of the first norm would be:

```
(:action check-call_reinf
  :parameters ( )
  :precondition (and (not (checked-call_reinf)))
  :effect (and
   (when (and (active_call_reinf) (not (or (and (not (true_predicate)))))) (active_call_reinf))
   (when (and (or (and (some_door_open)))
     (not (or (and (not (true_predicate)))))) (active_call_reinf))
   (when (or (and (not (true_predicate)))) (not (active_call_reinf)))
   (when (and (not (violation_point_call_reinf)) (active_call_reinf)
      (not (or (and (not (true_predicate))))) (not (or (and (not (alone))))))
      (violation_point_call_reinf))
   (when (and (not (violation_point_call_reinf)) (or (and (some_door_open)))
      (not (or (and (not (true_predicate))))) (not (or (and (not (alone))))))
      (violation_point_call_reinf))
   (when (violation_point_call_reinf) (not (violation_point_call_reinf)))
   (checked-call_reinf)))
```

Table 1. Belief Updates, Belief and Goals Transformation

2APL	PDDL	

`BeliefUpdates:`
 	
`{alone and equals(time, X)}`
`CallReinforcements()`
`{not alone, not equals(time, X),`
`equals(time, X+10)}`
	`(:action CallReinforcements`
	`:parameters ( )`
	`:precondition (and`
	`(checked-call_reinf)`
	`(checked-repair))`
	`:effect (and (not (alone))`
	`(not (checked-call_reinf))`
	`(not (checked-repair))`
	`(increase (time) 10)))`
		Domain

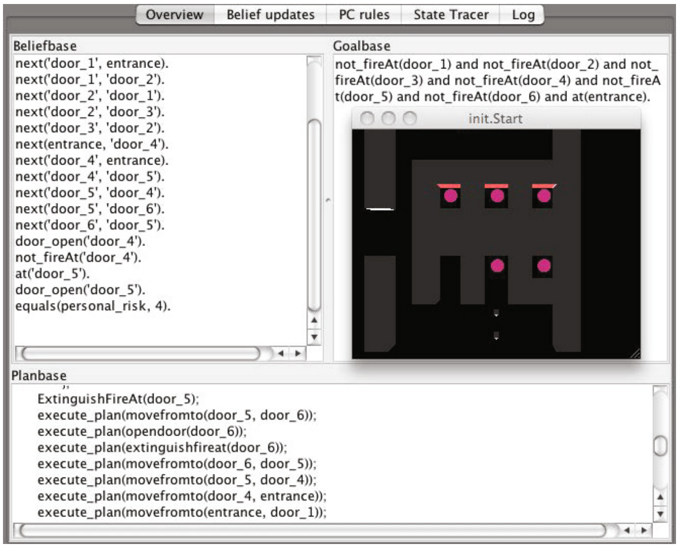
`Beliefs:`
 	
`true_predicate.`
`at(entrance).`
`alone.`
`equals(ranking, 3).`
`equals(personal_risk, 2).`
`equals(time, 0).`
`next(entrance, door_1).`
`next(door_1, door_2).`
`...`
	`(true_predicate)`
	`(at entrance)`
	`(alone)`
	`(= (ranking) 50)`
	`(= (personal_risk) 2)`
	`(= (time) 0)`
	`(next entrance door_1)`
	`(next door_1 door_2)`
	`...`
		Problem

`Goals:`
 	
`not_fireAt(door_1) and`
`not_fireAt(door_2) and`
`not_fireAt(door_3) and`
`not_fireAt(door_4) and`
`not_fireAt(door_5) and`
`not_fireAt(door_6) and`
`at(entrance)`
	`(and (not (active_repair))`
	`(checked-call_reinf)`
	`(checked-repair)`
	`(not_fireAt door_1)`
	`(not_fireAt door_2)`
	`(not_fireAt door_3)`
	`(not_fireAt door_4)`
	`(not_fireAt door_5)`
	`(not_fireAt door_6)`
	`(at entrance))`	

The belief updates (see section 3) of the 2APL agent (where the first part is the precondition, the second is the name and the third is the postcondition) can be considered the capabilities (actions) of the agent. These will be transformed to a PDDL domain action. More belief updates of the agent might be *MoveFromTo*, *LowerRanking*, *OpenDoor* and *ExtinguishFireAt*. The beliefs of the agent are transformed to the PDDL problem instance. The beliefs include the initial values of the factors time, danger and ranking. Finally, the goals of the agent are directly transformed to the goals in the PDDL problem instance[2]. To the goals are added the normative objectives explained in subsection 5.3, that is, that there is no active obligation coming from violated norms at the end of the execution of the plan. An example of belief updates, beliefs and goals in 2APL and how they are translated in PDDL domain and problem can be seen in Table 1.

7 Experimental Setup and Results

In this section we present the experimental setup and the results that occur from the simulation of the example in Section 2.



Beliefbase
```
next('door_1', entrance).
next('door_1', 'door_2').
next('door_2', 'door_1').
next('door_2', 'door_3').
next('door_3', 'door_2').
next(entrance, 'door_4').
next('door_4', entrance).
next('door_4', 'door_5').
next('door_5', 'door_4').
next('door_5', 'door_6').
next('door_6', 'door_5').
door_open('door_4').
not_fireAt('door_4').
at('door_5').
door_open('door_5').
equals(personal_risk, 4).
```

Goalbase
```
not_fireAt(door_1) and not_fireAt(door_2) and not_
fireAt(door_3) and not_fireAt(door_4) and not_fireA
t(door_5) and not_fireAt(door_6) and at(entrance).
```

Planbase
```
    ExtinguishFireAt(door_5);
    execute_plan(movefromto(door_5, door_6));
    execute_plan(opendoor(door_6));
    execute_plan(extinguishfireat(door_6));
    execute_plan(movefromto(door_6, door_5));
    execute_plan(movefromto(door_5, door_4));
    execute_plan(movefromto(door_4, entrance));
    execute_plan(movefromto(entrance, door_1));
```

Fig. 5. 2APL Agent and Environment

Having described the fire emergency use case in Section 2 we deploy a set of scenarios (the graphical result of the simulation is as in Figure 5). While the objective of the agent is to extinguish the fires and get to the entrance of the building, the agent needs to take into consideration the normative commitment of not being alone when doors are opened. The three factors (time, risk, ranking) play an important role. The planner has to find a plan that either conforms to the norm, or, in the case it does not, to make sure that the appropriate steps are taken, in order to satisfy the repair norm that is activated with the violation of the first (have the ranking lowered).

We assume that the agent's welfare lies equally on the three factors. We remind that the factors get affected (increased or decreased) by the execution of the actions as explained in Section 2. As a result the formula $(personal\_risk + time - ranking)$

[2] The details of the transformation are not presented in the paper due to limitation of space.

can serve as an appropriate metric for the normative planner to produce the most beneficial plan. Trying the model under different setups of the three factors, the simulation produces two distinct results.

Whenever the weight of the waiting time for the reinforcements to arrive is considerably small compared to and the estimated risk for opening the door when being alone and the estimated loss of ranking in case of violating the norm, the agent decides to comply with the norm, call and wait for reinforcements. An example of such a setup can be seen in Table 2 and the resulting plan in Figure 6.

Table 2. Setup leading to compliance to the norm

action	personal_risk	time	ranking
CallReinforcements	0	+10	0
MoveFromTo	+1	+1	0
OpenDoor			
{if alone}	+3	+1	0
{if not alone}	+1	+1	0
ExtinguishFireAt	0	+2	0
LowerRanking	0	0	-20

Table 3. Setup leading to violation to the norm

action	personal_risk	time	ranking
CallReinforcements	0	+12	0
MoveFromTo	+1	+1	0
OpenDoor			
{if alone}	+25	+1	0
{if not alone}	+1	+1	0
ExtinguishFireAt	0	+2	0
LowerRanking	0	0	-11

```
Plan 1

callreinforcements
movefromto(entrance ,door_4)
opendoor(door_4)
extinguishfireat(door_4)
movefromto(door_4 ,door_5)
opendoor(door_5)
extinguishfireat(door_5)
movefromto(door_5 ,door_6)
opendoor(door_6)
extinguishfireat(door_6)
movefromto(door_6 ,door_5)
movefromto(door_5 ,door_4)
movefromto(door_4 ,entrance)
movefromto(entrance ,door_1)
opendoor(door_1)
extinguishfireat(door_1)
movefromto(door_1 ,door_2)
opendoor(door_2)
extinguishfireat(door_2)
movefromto(door_2 ,door_3)
opendoor(door_3)
extinguishfireat(door_3)
movefromto(door_3 ,door_2)
movefromto(door_2 ,door_1)
movefromto(door_1 ,entrance)
```

Fig. 6. Conforming to norm. Calling reinforcements and entering the building accompanied.

```
Plan 2

lowerranking
movefromto(entrance ,door_1)
opendoor(door_1)
extinguishfireat(door_1)
movefromto(door_1 ,door_2)
opendoor(door_2)
extinguishfireat(door_2)
movefromto(door_2 ,door_3)
opendoor(door_3)
extinguishfireat(door_3)
movefromto(door_3 ,door_2)
movefromto(door_2 ,door_1)
movefromto(door_1 ,entrance)
movefromto(entrance ,door_4)
opendoor(door_4)
extinguishfireat(door_4)
movefromto(door_4 ,door_5)
opendoor(door_5)
extinguishfireat(door_5)
movefromto(door_5 ,door_6)
opendoor(door_6)
extinguishfireat(door_6)
movefromto(door_6 ,door_5)
movefromto(door_5 ,door_4)
movefromto(door_4 ,entrance)
```

Fig. 7. Violating norm. Entering the building alone and repairing by lowering ranking.

Whenever weighing the same factors gives a contrary result, the agent decides not to wait and start the fire battle alone, having the ranking lowered as a consequence. An example of such a setup can be seen in Table 3 and the resulting plan in Figure 7.

Metric-FF planner [6] allows to choose between the weight of the quality of the search and the metric of the plan. Giving more weight to minimise search cost, the planner is able to find a solution in less than a second, but the resulting plan usually differs from the best options. Giving more weight to the plan quality, the planner is able to find a solution in a minute or more. For the example above, if we give the same relevance to speed and plan quality, the planner provides a solution in less than 20 seconds.

We used this example and made several experiments, expanding every time the domain (the number of doors in the building and thus the number of steps the agent needs to take in order to extinguish the fires behind them). The experiments were done on Mac OS version 10.8 with an Intel processor 2.9GHz Core i7. Increasing the importance of the metric in the planning process (quality/metric) becomes rather costly. Nevertheless, the quality of the plans returned when executing with small plan metric importance seems to be good and we get coherent plans depending on the values of the cost parameters time/personal-risk/ranking. We provide execution times for both possible plans that might occur, depending on the setting of the parameters (the one where the agent obeys the norm and calls for reinforcements and the other where the agent violates it and enters the building alone). The results can be seen in Table 4.

Table 4. Experimental Results

Results			
door number	**quality/metric**	**norm status**	**time (sec)**
6	5/1	not violated	0.00
	5/1	violated	0.00
12	5/1	not violated	0.01
	5/1	violated	0.01
14	5/1	not violated	0.01
	5/1	violated	0.01
6	5/2	not violated	0.00
	5/2	violated	0.05
12	5/2	not violated	0.01
	5/2	violated	4.19
14	5/2	not violated	0.01
	5/2	violated	37.29
6	5/3	not violated	0.01
	5/3	violated	0.05
12	5/3	not violated	0.16
	5/3	violated	∞
14	5/3	not violated	0.35
	5/3	violated	∞

We are currently working on further improvements in our model and in the Metric-FF settings in order to reduce the time needed while keeping an acceptable accuracy in the results given.

8 Discussion and Related Work

Our work stems from the fact that while regulation of agent's behaviour has become a necessity in current multi-agent environments, little work on practical reasoning mechanisms within normative environments exists.

In the planning domain, the closest approach to ours is the introduction of control rules in PDDL planners. Control rules [1] are formulas expressed in Linear Temporal Logic (LTL) which, applied to the (forward chaining) search, they reduce the search space by pruning paths that do not comply to the rules. More recently, the PDDL 3.0 specification [5] has added strong and soft constraints (expressed in LTL formulas) which are imposed on plan trajectories, as well as strong and soft problem goals, which are imposed on a plan. However in order to use control rules or PDDL 3.0 constraints, we would need to reduce all the operational semantics in our norm lifecycle into LTL formulas, which would imply merging all possible future temporal paths into a single future path, loosing expressivity. PDDL 3.0 also lacks the operator "until", which would permit us to express some properties in our norm lifecycle (ex. a norm is violated when activated at some point and maintenance condition does not hold at some state after this and deactivating condition does not hold at any state in between). Furthermore, in our model a norm can be activated and deactivated (and possibly violated) several times during the execution of a plan, something not possible to be expressed directly in PDDL 3.0.

Close to our work is also [8]. It presents the NoA (normative agent) architecture, comprising the NoA language for the specifications of plans, norms and contracts, and the NoA architecture, which operates as an interpreter and executor of such specifications and represents a concrete implementation of an approach to norm-governed practical reasoning. The plans get instantiated at runtime according to whether they can satisfy a norm and they are labelled as consistent or inconsistent according to the currently activated permissions and prohibitions. With this labelling mechanism, the deliberation process becomes informed about possible norm violations. While the semantics used for the specification of the norms and a norm's activation and deactivation as well as plans specification are very similar to the ones we use, the approach differs in that it uses sets of predefined plans in order to achieve tasks or states.

In [9] the authors extend a BDI agent language, enabling them to enact behaviour modification at runtime in response to newly accepted norms. This consists of creating new plans (adding them to the plan library) to comply with obligations and suppressing the execution of existing plans (removing them from the plan library) that violate prohibitions. They demonstrate the approach through an implementation of in the AgentSpeak(L) language [12].

In a more recent work, Modgil et al. [10] propose an architecture for monitoring norm-governed systems. The system deploys monitors that receive inputs from trusted observers, and processes these inputs together with *Augmented Transition Networks* (ATNs) representations of individual norms. In this way, monitors determine the states of interest relevant to the activation, fulfilment, violation and expiration of norms. Additionally, the monitoring system is corrective in the sense that it allows norm violations to be detected and reacting to them.

In [3] the authors provide an overview of the issues encountered in implementing different norm aspects in agents. Using a simulation scenario they implement rules that lead to the adoption or not of a norm, the generation of norm compliant plans and the monitoring of a norm enforcement. However they provide hard coded mechanisms for dealing with norms (norms are integrated into the agent's code) and as a result several implementation complications occur.

Recent work detailed in [7] assumes the workflows/plans to exist already, but describes how to cope with deviations/exceptions. Our approach differs in that we merge the planning and evaluation with respect to the norms process into one and build appropriate plans during runtime phase.

The strength of our approach lies on the effort made to keep as much flexibility as possible in the norm influence to the decision making while keeping it practical. One benefit of our approach is that normative states can be defined separately to the agent's capabilities, allowing in this way to express "what" is desired or undesired but not necessarily "how" this should be achieved or avoided. Emphasis is given on the agent's welfare attributes and how these are influenced by the execution of a plan with respect to a set of norms. Given the fact that two agents might vary in the importance that they give to individual factors, different outcomes concerning the achievement of a goal are expected. Referring to the firefighter example, in the case of a upright agent, he might call and await for the reinforcements, knowing that they might take long to arrive, while in the case of a less committed agent, he might risk his loyalism in order to act faster. Further to this, multiple norms may be in conflict and an agent must make informed choices. For example, if an agent has an obligation to achieve a goal g, but all the methods / plans that it possesses to achieve goal g will violate one prohibition or another, then it is confronted with a conflict that requires the agent to make a decision on which prohibition to violate to achieve g (or not to fulfill its obligation at all). In such a case, the agent being aware of the consequences that the violation of each one could have given the circumstances, might be able to make an informed decision based on situational criteria rather than always producing the same outcome.

A main drawback is that whereas our framework works well in environments where the consequences of an agent's conformance to normative restrictions are known or can be estimated, it is not always the case that this is feasible. In many cases the effect of a violation cannot be known in advance, or might even be non existing (not always does crossing a red traffic light implies a fine). Our approach assumes that there is an instant global enforcing mechanism and cannot for the moment handle probabilistic effects of norm deviation. While we cannot claim that our approach covers all aspects of normative multi-agent environments, we consider it a first step that could have further extensions towards more advanced implementation.

Our objective is to extend our work towards multiple agent decision making. While one agent considers its personal settings in order to create a plan that satisfies its needs, the issue becomes more complex when dealing with more agents and social metrics. Such extension might become subject to many questions interesting to be studied, such as how can the agents share their preferences about plans, how to manage asymmetrical influence of the norms to different agents, or how to balance the collective preferences and costs with the individuals' when comparing among several plans' adequacy.

Acknowledgments. This work has been supported by the project ICT-FP7-289067 SUPERHUB.

References

1. Bacchus, F., Kabanza, F.: Using temporal logics to express search control knowledge for planning. Artificial Intelligence 116(1-2), 123–191 (2000)
2. Dastani, M.: 2APL: a practical agent programming language. JAAMAS 16(3), 214–248 (2008); Special Issue on Computational Logic-based Agents
3. Dignum, F., Aldewereld, H., Vanhee, L.: Implementing Norms? COIN@WI-IAT (2011)
4. Fikes, R., Nilsson, N.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In: Artificial Intelligence, vol. 2, pp. 189–208. Elsevier Science Publishers Ltd., Essex (1971)
5. Gerevini, A., Long, D.: Plan constraints and preferences in PDDL3. In: 5th International Planning Competition (2006)
6. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14, 253–302 (2001)
7. Lam, J., Guerin, F., Vasconcelos, W., Norman, T.J.: Building Multi-Agent Systems for Workflow Enactment and Exception Handling. In: Padget, J., Artikis, A., Vasconcelos, W., Stathis, K., da Silva, V.T., Matson, E., Polleres, A. (eds.) COIN 2009. LNCS (LNAI), vol. 6069, pp. 53–69. Springer, Heidelberg (2010)
8. Kollingbaum, M.J.: Norm-governed Practical Reasoning Agents. Tech. Rep. (2005)
9. Meneguzzi, F., Luck, M.: Norm-based behaviour modification in BDI agents. In: 8th International Conference on Autonomous Agents and Multiagent Systems (2009)
10. Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., Luck, M.: A framework for monitoring agent-based normative systems. International Foundation for Autonomous Agents and Multiagent Systems (May 2009)
11. Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a Formalisation of Electronic Contracting Environments. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 156–171. Springer, Heidelberg (2009)
12. Rao, A.S.: Agentspeak(l): Bdi agents speak out in a logical computable language. In: 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Agents Breaking Away, pp. 42–55. Springer (1996)
13. Panagiotidi, S., Vázquez-Salceda, J.: Normative Planning: Semantics and Implementation. In: 13th International Workshop on Coordination, Organizations, Institutions and Norms in Agent Systems (COIN@WI-IAT), Lyon, France (2011)

An Agent-Based Simulation Approach to Comparative Analysis of Enforcement Mechanisms

Tina Balke[1,2], Marina De Vos[2], and Julian Padget[2]

[1] University of Surrey, Centre for Research in Social Simulation
t.balke@surrey.ac.uk
[2] University of Bath, Dept. of Computer Science
{mdv,jap}@cs.bath.ac.uk

Abstract. Incentive-based enforcement can be an effective mechanism for fostering cooperation in open distributed systems. The strength of such systems is the absence of a central controlling instance, but at the same time, they do depend upon (voluntary) regulation to achieve system goals, creating a potential "tragedy of the commons". Many different mechanisms have been proposed, both in the multi-agent systems and the social science communities, to solve the commons problem by using incentive-based enforcement. This paper advocates the use of agent-based simulation to carry out detailed comparative analysis of competing enforcement mechanisms, by providing common settings, the environment and the basis for comprehensive statistical analysis. To advance this argument, we take the case study of wireless mobile grids, a future generation mobile phone concept, to ground our experiments and analyse three different enforcement approaches: police entities, image information and a well-known existing reputation mechanism. The contribution of this paper is not the enforcement mechanisms themselves, but their comparison in a common setting through which we demonstrate by simulation and statistical analysis that enforcement can improve cooperation and that a relatively small percentage (of the population as a whole) of police agents outperforms (under the chosen metrics) image- and reputation-based approaches. Hence, qualified conclusions may be drawn for the application of such mechanisms generally in open distributed systems.

1 Introduction

Open distributed systems allow autonomous entities with some form of social relationship to join and leave freely as well as to perform actions such as interacting with other entities. Entities base their decisions and actions on their own goals as well as their expectations about the system and the behaviour of the other entities. The result of the combined individual decisions and actions is a global emergent behaviour that—in contrast to the individual decision making processes—can be perceived from outside the system.

The principal advantage and disadvantage of open distributed systems is that at design-time, it is unknown precisely what individual and collective behaviour may be exhibited by participating entities. Complete control of even closed distributed systems has proven a very challenging problem. Rigid control of open systems, especially given their increasingly pervasive nature, is unrealistic; not only is imposition of controls a reaction to a perceived threat (to system integrity), it also fails to recognize open systems as a nascent opportunity.

One particular problem in open distributed systems (be it relay-routing, peer-to-peer, cloud computing, etc.) is that they require some form of contribution on the part of participants, which translates into some form of cost to them. Participants can exhibit strategic behaviour and are not necessarily cooperating (i.e. contributing to the system). For an agent, making resources available therefore has the danger that its good behaviour is not reciprocated, resulting in no inherent value in cooperation for a participant. A lone cooperating user draws no benefit from their cooperation, even if the rest of the system might. Guaranteed cost paired with uncertainty or even lack of any resulting benefit does not induce cooperation for a utility-maximizing user. Without any further incentives, rational users therefore would not cooperate in such an environment and all will be worse off than if they cooperated. This phenomenon is referred to as the "Tragedy of the Commons" [14,20]. Ostrom's work analyses how contributions in such (semi-)closed systems can be reinforced through mechanisms such as identity and sanctions such as ostracism. The problem with the applicability of Ostrom's work in the above described settings is that Ostrom focuses on small (relatively closed) communities which are qualitatively different (easier identification of agents and easier application of ostracism) and might typically require quite different solutions. As a consequence of the above problem, and of the limitations on the extent to which rigid control is feasible in open distributed systems, enforcement mechanisms offer a means to reduce the prevalence of the commons phenomenon [15].

To evaluate enforcement mechanisms empirically, as we propose here, a suitable domain is needed, in order to be able to identify which aspects can be quantified and how. However this inevitably creates the risk that decisions are made, or metrics are constructed that are domain-specific. The domain chosen in this paper is the wireless mobile grid (WMG), which is described in more detail in the next section. We do not make a judgement as to whether the WMG concept is viable or not: it is simply a novel example of the kind of emerging 'digital commons' that makes it suitable as a case study for which it would also be useful to get some early indicators of which kinds of enforcement are effective and what the associated costs might be.

The WMG, as an opportunistic network made possible by chance co-location, exhibits many of the characteristics of an open system: participants are free to join or leave at any time, identity is not authenticated and free-riding appears to be easy. Consequently, repeat encounters are likely to be few and participant turnover high. Thus, the participant contributions required to sustain it may be difficult to acquire or to incentivize. The relative complexity of the scenario, at least until better understood, makes an analytical or game-theoretic approach infeasible at this stage, so we advocate agent-based simulation as a means to establish a better understanding of the dynamics and to evaluate side-by-side three well-established enforcement mechanisms. Based on the common setting provided by the simulation environment we examine the mechanisms' advantages and disadvantages with respect of one another. For this purpose, in the next section, we describe the case study, then in Section 3 we present the three enforcement mechanisms to be compared. The simulation experiments which were validated with the help of domain experts and its results are discussed in Sections 4 and 5. This paper ends with a short summary of the findings as well as a discussion of their implications for open distributed systems (Section 7).

2 The Wireless Mobile Grid Case Study

To demonstrate the use of agent-based simulation for the comparative analysis of enforcement mechanisms, we start by establishing a common case study for our experiments which portrays the particular features of open distributed systems sketched in the previous section. In one sense, the domain details of the case study are not especially important, but are simply there to ground the scenario, rather than using an abstract scenario which can be harder to assimilate. Thus, the particular domain is the so-called "wireless mobile grid" (WMG); a mechanism proposed by Fitzek and Katz [10] to address the energy issues inherent in 4th generation mobile phones. This paper is not about the plausibility or otherwise of WMGs, but about the comparative effectiveness of different approaches to enforcement in the context generated by WMGs, representing an instance of the broader class of open distributed systems.

In WMGs, as well as using the traditional 3G (or LTE) communication link with base stations, users are envisioned as sharing resources in a peer-to-peer fashion using a short-link connection protocol such as IEEE802.11 WLAN. The advantage of this short-link connection is that it uses less power and allows for higher data rates. However, in order to function properly WMGs require collaboration between users, which may be difficult to realize. The main problem in WMG is that collaboration comes at the cost of battery consumption[1]. In consequence, rational users will prefer to receive the resources without any commitment to contribute themselves. However, if a substantial number of users follow this selfish strategy, the WMG will not work and none will benefit from the potential energy savings arising from cooperation [26]. A WMG is an interesting and novel example of an open distributes system, in which there are autonomous users with their own goals who can freely join and leave, who can interact with one another over short-range connections for short periods, and whose individual actions contribute to the success or failure of the WMG as a whole.

Purely technical (hard-ware or hard-coded) solutions for ensuring non-compliance in open distributed systems are frequently subverted (see [18] for example). Hence, the approach we take here, which is to employ enforcement mechanisms such as reputation information or police agents that regulate WMGs by social means.

3 Enforcement Mechanisms for Wireless Mobile Grids

Many possible enforcement mechanisms exist, so how to choose suitable ones? Balke and Villatoro [3] provide a systematic overview of possible enforcement options by identifying the roles actors can have in an enforcement setting and discussing all possible combinations of these roles in the enforcement process. We do not reproduce the details of the mechanisms analysed in [3], for sake of space as much as correctness,

[1] The patterns of collaboration and interaction in WMGs are described in details in Section 4.1.

but draw on their conclusions to select three mechanisms to concentrate on in more detail: reputation information, image information, also known as direct trust, and police agents. We choose these because of their popularity in the agent community as much as for their complementary foundational concepts that allows us to explore a wider range of options.

3.1 Utilization of Police Entities

The utilization of police entities can be thought of as the implementation of entities with normative power [16], created for the enforcement purpose by the owner of the system that participate in the system (in our example, the WMG), and have permission from the owner to punish, if detected, negative/inappropriate behaviour (i.e. non-compliance) by means of sanctions. In contrast to regimentation (i.e. complete control) [7], the police entities do not control all actions but only act as enforcers when violations are detected. Detection of violations is done by the police entities themselves, who test the behaviour of entities and react to what they detect. Several kinds of sanctions can be imagined depending on the severity of the non-compliance, such as complete exclusion from the WMG or penalty payments either monetary or in terms of energy.

3.2 Image Information

Image information [19], also called direct trust, is a global or averaged evaluation of a given agent – usually called the target – on the part of an individual. It consists of a set of evaluative beliefs about the characteristics of a target. These evaluative beliefs concern the ability or the possibility of the target to fulfil one or more of the evaluator's goals, e.g. to cooperate in a WMG transaction. An image basically gives the evaluator's opinion of whether the target is "good" or "bad" or "not so bad" etc. with respect to a norm, a standard, a skill etc. When utilizing image information, an agent uses *its own* information about the past behaviour of the potential interaction partner and makes decisions based on this information.

3.3 Reputation Information

Reputation information, in contrast to image information, comprises not only agent' own acquired image information, but that obtained from other agents as well. Thus, reputation in this paper is understood as the process of and the effect of the transmission of a target's image. In contrast to image information alone, as described above, when images are circulated more information becomes available to the individual agents. However, the circulation of information itself can generate costs. Furthermore it is possible that agents may circulate false image information to increase their value relative to other agents.

4 The Simulation Design

Having briefly outlined the enforcement mechanisms under examination, we now present the basic simulation setup. We first describe the agents and their decision making behaviour and then outline how the enforcement mechanisms have been implemented. Concerning the technical components of the wireless mobile grid, we adopt the well-established "flat earth model" [17] that assumes symmetry (i.e. if node A can hear node B, B can hear A) and an absence of obstacles that might reduce transmission quality. The flat-earth model is a widely accepted simplification made in the mobile communications community and has been used for simulation presented in mobile communication centred articles on this topic (e.g. [2]). Furthermore we assume that all agents have identical mobile phones, for which we use the energy consumption profile data reported in [22]. A reason for this assumption is that [22] explains that the Nokia N95 mobile phone is a representative phone with features for WMG communication and that the differences between different mobile phones are only marginal.

4.1 The Basic Agent Decision Process

The simulation uses one agent for each user/mobile phone pair. These agents move randomly in the simulation space and at any given point of time can interact with the agents that are within their (modelled) WLAN range. The agents make decisions that maximize their utility under the constraint of bounded rationality. Different agents are given different utility valuations. We define three kinds of non-police agents according to the behaviours for which they maximise: (i) "utility agents" that try to minimize battery consumption and avoid punishment (ii) "honest agents" that cooperate whenever possible, and (iii) "malicious agents" that try to undermine the system regardless of cost The agent's decision-making is based on incomplete knowledge of the system state, so they can only optimize for local utility, which may be different from the global utility. Local knowledge is determined by two factors: the agent's location and its WLAN radius. A full Cartesian model is unnecessary, since we only need to model proximity, hence an agent location is modelled as $l \in \mathbb{R} \bmod 1$, that is the interval wraps around. An agent at ϵ and another at $1-\epsilon$ are 2ϵ apart. The proximity of two agents is determined by each agent's WLAN radius (r_v). An agent at l_1 has radius $[l_1 - r_{v1}, l_1 + r_{v1}]$ and another at l_2 has radius $[l_2 - r_{v2}, l_2 + r_{v2}]$. Communication between these two is possible if these intervals intersect.

The procedure for the agent's decision making process and its utility considerations (see Figure 1) are based on the following issues:

1. Each agent has the task of acquiring a whole file through downloading (over 3G) or exchanging (over WLAN) file chunks. The agent must decide whether to download it all or to search for a collaboration partner with whom to share the work. File size is the first determinant: if the file is small and the potential costs of finding a collaborator are higher than the potential gains, then the agent will download all the chunks itself. Otherwise, it looks for nearby agents.
2. If the neighbourhood is sparsely populated, then the chances of finding sufficient partners is low and the agent downloads all the chunks itself. Otherwise, it sends a cooperation request specifying the file whose chunks it requires using WLAN broadcast.
3. If the agent receives a cooperation request for the same file, there is no need to send a request, so it just replies using WLAN broadcast.

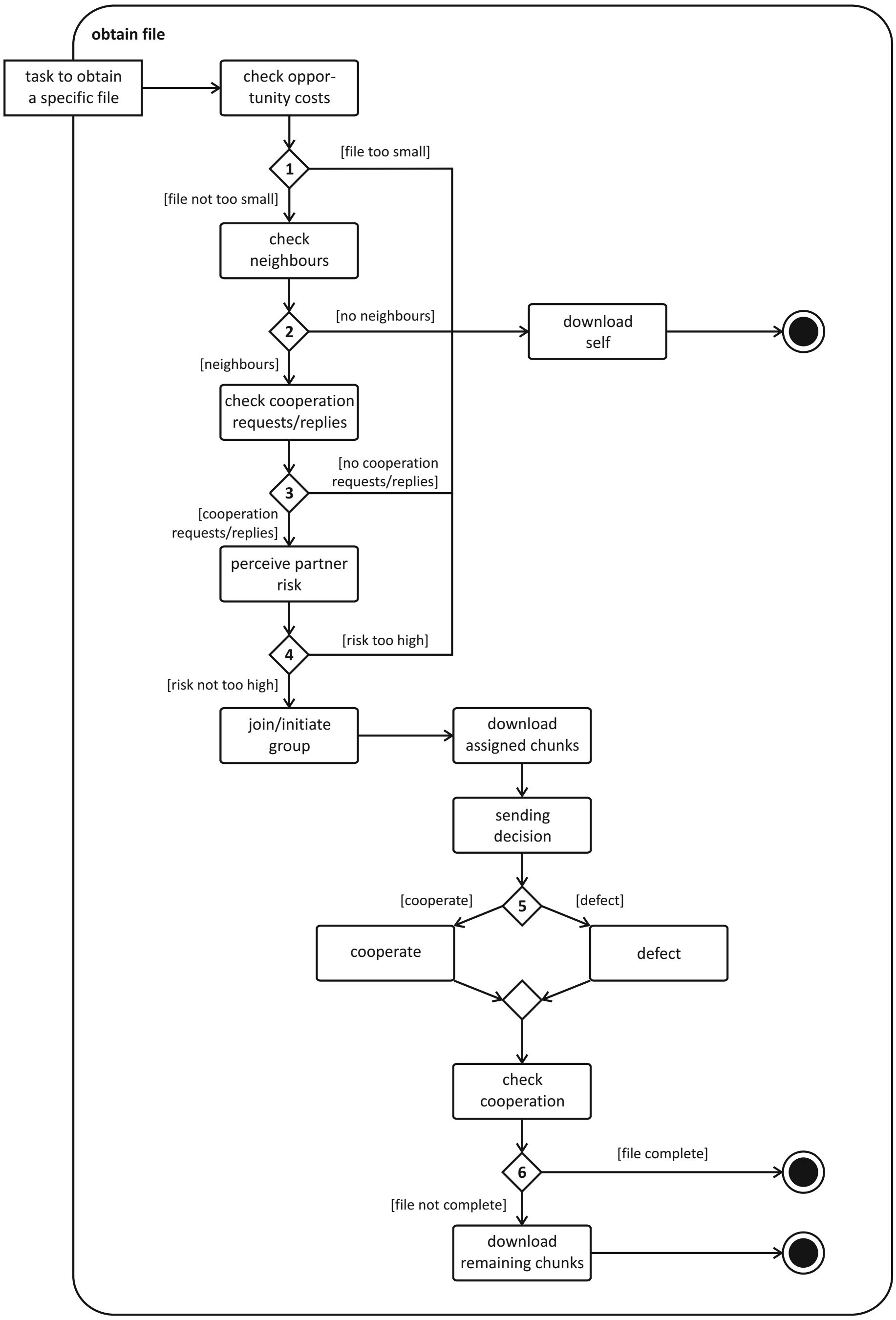


Fig. 1. Download considerations

4. Having sent a cooperation request, the agent awaits responses. From the positive replies, the agent selects collaboration partners, possibly using image or reputation mechanisms to decide.
5. Once a cooperation group has been formed, an agent has to decide: (i) whether to download its promised chunk(s) (over 3G) and (ii) whether to share its chunks with the cooperation group.
6. The cooperation decision depends on the agent's individual utilities for cooperation and defection. Thus, an utility agent (see definitions above) compares energy benefits from defecting now, against the future costs arising from detection in terms of the likelihood and level of a fine, by comparing the number of past defections with the number and level of past fines. An honest agent will always cooperate and will never defect. A malicious agent, in contrast, will always defect.

Having made its decision, the last step is to wait and see whether the cooperation partners send their promised shares. For missing shares, the agent repeats the decision process outlined above.

4.2 Implementing the Enforcement Mechanisms

We now describe the implementation of the three enforcement mechanisms.

Enforcement Agents. A police agent has the same properties as an ordinary agents, except for restricted behaviour in that it: (i) responds positively to cooperation requests, if not already committed, and then performs its share of downloading and sharing, (ii) does not have its own file download tasks set at the beginning and (iii) does not send cooperation requests.

A police agent never defects. Its energy costs count towards the total WMG energy costs – so that enforcement costs are included. It also monitors cooperation by checking on the number of defections in the cooperation group to which it belongs and fines those that have defected. The fine is measured in terms of energy and is set at three times the agent's relative gain from defection[2].

Enforcers make no distinction between intention and absence of action: it only matters whether an agent kept the cooperation agreement by sending its chunks by the cooperation deadline. The same defection may be observed by more than one enforcer, but the fining mechanism ensures that an agent is not fined twice for the same offence.

Image Information. Image information [5] is used by agents in making cooperation decisions. No police agents are used. Image information is information acquired about another agent through direct interaction with that agent. Thus, the experience of *own* past interactions is used to evaluate new cooperation requests. If the image is positive, the collaboration proceeds. Each agent could have an individual defection tolerance level for all its past collaborators, but to keep the range of experiments within what can be reported in this paper, this is uniformly set to zero. Thus, the object of a defection

[2] We tested a range of alternative fine levels. Space limits prevent a full report, but the value of three time's the gain exhibited the best balance between deterrent and the fine not being disproportionate to the offence.

will never collaborate with the defector again. If an agent has no image for the originator of a cooperation request, it treats the image as positive.

The next step is to incorporate image information in the utility computation. First, the agent computes the fraction of the number of times its cooperation (request and offer) was rejected because of its image. This value is then weighted by (i) the chunk size and (ii) the cooperation group size.

The latter is significant because, the larger the group, the greater the spread of negative image information if the agents defects.

The advantage of image information is its reliability. However, one problem often associated with image mechanisms is that an agent first needs its own experience to construct image information. Consequently, it can always be the object of a defection at least once. Reputation mechanisms are proposed as a way to avoid this problem. We explore this approach next.

Reputation Information. In reputation mechanisms, the image information of individual agents is circulated. A large number of reputation mechanisms have been put forward in the literature covering a range of circumstances. To select one suitable for WMG, we start by examining the requirements and constraints.

Although the work reported here is simulation-based, in a real WMG, the actual agents will include humans. This suggests any mechanism should allow for the subjective expression of trust based on individuals' perceptions. Additionally, the mechanism needs to be compatible with the non-numerical and non-monotonic models of human expression. Finally, the mechanism must be able to handle incorrect information, taking the sources of (reputation) information into account and identifying those that provide false information. Consequently, we consider three candidate mechanisms: Regret, Fire and Abdul-Rahman and Hailes (ARH), which are analysed in detail in [24]. Regret seems unsuitable because by default it requires a large number of messages to be sent (accounting for witness, neighbourhood and system reputation) which inevitably increases overall energy consumption. Fire is also unsuitable because of its basic assumptions that agents (i) willingly share their experience and (ii) report truthfully when exchanging information with one another. This leaves the ARH reputation mechanism [1], which fortunately satisfies our requirements.

ARH requires that each agent maintains a database of trust relationships that they use for themselves or to respond to the requests of others. The data is segregated into direct (image) and indirect (reputation) information. ARH defines a "trust-relationship" as a vectored connection between exactly two entities, which in some circumstances can be transitive. In this way they distinguish between direct trust relationships ("Alice trusts Bob.") and recommender trust relationships ("Alice trusts Bob's recommendations about the trustworthiness of other agents"). This allows entities to account for the source of reputation information as well as collecting and evaluating information about the reliability of recommenders. Another interesting contrast to other formalizations is that, reflecting the qualitative nature of trust, ARH does not use probability values or the $[-1, 1]$ interval, but a multi-context recording model with abstract trust categories that are easier for humans to understand. These trust values relate to certain contextual information ("Alice trusts Bob, concerning "table"-transactions. However, she does not trust him when it comes to "chair"-transactions.").

ARH models trust as context-dependent, so it is defined as a "troika" of (*agent-ID*, *Trust-Category*, *Trust-Value*), with trust categories such as "cooperation partner" or "recommender". ARH sets out a recommendation protocol for handling recommendation requests, statements and enquiries. A recommendation request is forwarded until one or more agents are found that can give information for the requested category and which is trusted by the penultimate agent in the chain. We do not support routing in the WMG simulation and implementing the protocol described above would result in large amounts of network traffic, impacting significantly upon the potential benefits of WMG. Thus, we simplify this aspect of ARH in such a way that an agent seeking a recommendation about a target sends out *one* broadcast message to its neighbours. If it receives no answers, the agent does not wait for further information, but as in the case of image information, cooperates with the target.

An agent, of whichever kind, uses the reputation mechanisms as follows:

1. If it has image or reputation information about the potential cooperation partner, it uses it.
2. If not, it sends a request for recommendations to its neighbours:
   (a) If there are no replies, the agent agrees to collaborate and will update its image information in due course in respect of the outcome of the collaboration.
   (b) If there are one or more replies, they are categorized by source into trusted, untrusted and unknown:
      i. Trusted source: the agent updates its local reputation information using the most trusted source and acts accordingly (i.e. positive recommendation: collaborate, negative: not). For equally trusted sources, the first reply is used.
      ii. Untrusted source: the information is kept for later validation but not taken into account for the current decision.
      iii. Unknown source: information is treated as for a trusted source.

Different kinds of agent respond differently to reputation requests. The utility maximizing agent will not send any information, because answering a message costs energy. An honest or a malicious agent answers on average one request per interaction event, in order to limit energy spent on answering reputation requests. An honest agent always reports truthfully about the target, with the aim of improving the overall information level in the system. A malicious agent however, if the target is not itself, always gives negative feedback on the target, with the aim of enhancing its relative reputation.

5 Hypotheses, Parameters and Results

5.1 Simulation Hypotheses and Parameters

As pointed out in the introduction this paper focuses on the use of agent-based simulation to carry out a detailed comparative analysis of competing mechanisms (i.e. enforcement mechanisms in our scenario) and to determine which of these meets the system objective (which we defined as energy saving) best. To test the impact of the three different enforcement mechanisms on the cooperation problem and the resulting energy consumption in WMGs, we formulate the following hypothesis:

Table 1. Simulation Variables

Name	Range/Type	Simulation Parameter
Number of Agents ($\mid \mathcal{A} \mid$)	$[2, \infty]$	200, 400, 800
Utility Agents as % of $\mid \mathcal{A} \mid$	[0,100]	0, 25, 50, 75, 100
Malicious Agents as % of $\mid \mathcal{A} \mid$	[0,100]	0, 25, 50, 75
Honest Agents as % of $\mid \mathcal{A} \mid$	[0,100]	0, 25, 50, 75
Enforcement Mechanism		None, Police Agents, Image Info., Reputation Info.
Number of Police Agents $\mid \mathcal{A}_{\text{Enf}} \mid$ as % of $\mid \mathcal{A} \mid$	$[0,\infty]$	0.5, 1, 2, 3, 5 % of $\mid \mathcal{A} \mid$, s.t. $\mid \mathcal{A}_{\text{Enf}} \mid > 1$
$\rho_{neighbourhood}$	$[0, \mid \mathcal{A} \mid -1]$	10, 20

Hypothesis 1: The presence of an enforcement mechanisms reduces the average energy consumption compared to when there is none.

We can assume that an enforcement mechanism has an effect on energy consumption, but if it affects cooperation – and in consequence energy consumption – is this effect constant across various simulation settings? Specifically, we want to establish sensitivity across a range of parameters: (i) population size, $\mid \mathcal{A} \mid$ (ii) population density (the average number of agents within each others' WLAN radius), $\rho_{neighbourhood}$, and (iii) population composition, that is proportions of utility, honest and malicious agents.

To test the influence of $\mid \mathcal{A} \mid$ and whether either of the other two parameters affects the simulation results, we check the null-hypotheses that no difference in simulation results can be observed when these parameters are varied. We then examine what impacts upon the different enforcement mechanisms:

Hypothesis 2: The success (in terms of the average energy consumption) of a WMG using reputation-based enforcement depends on population size, density and composition.

Hypothesis 3: The success (measured by average energy consumption) of a WMG using police agents as the enforcement mechanism depends on population size, density and composition as well as the number of police agents $\mid \mathcal{A}_{\text{Enf}} \mid$.

Hypothesis 4: The success (measured by average energy consumption) of a WMG using image-based enforcement depends on population size, density and composition.

For all of the above, we use the experiment configuration shown in Table 1, which summarizes the factorial experiments performed and the values over which each simulation parameter ranges.

5.2 Simulation Results

The experiments consist of 50 runs for each of the 468 parameter combinations in Table 1, making 23,400 runs in total. We used ANOVA to test the significance relationship between the independent variables (the parameters in the simulation) and the dependant

Table 2. Analysis of variance of experiments with and without enforcement

Source	**Sum of Squares**	**Degrees of Freedom**	**Mean Squares**	**F**	**Prob > F (= p-value)**
Enforcement	224.759	4	56.1898	1539.94	< 0.0001
Error	709.842	19454	0.0365		
Total	934.601	19458			

Table 3. Post-hoc analysis of variance of enforcement mechanisms

Source	**Sum of Squares**	**Degrees of Freedom**	**Mean Squares**	**F**	**Prob > F (= p-value)**
Image-Information	26.969	1	26.969	684.73	< 0.0001
Police Agents	168.95	5	33.7907	801.85	< 0.0001
Reputation	12.308	1	12.3078	332.01	< 0.0001

variables (the number and ratio of defections and energy consumption)[3]. We also applied Tukey's test as a post-hoc ANOVA, which identifies the impact of specific variables on the overall result.

Testing Hypothesis 1. We can now analyse the simulation results to test the hypotheses formulated in the previous section. First, we test hypothesis 1 and look at mean energy consumption when there are different enforcement mechanisms employed. By means of ANOVA, we can test whether there is sufficient evidence to reject the null hypothesis that enforcement mechanisms have no effect on energy consumption. Table 2 shows the results of this comparison.

As the significant p-value suggests, we can reject the null hypothesis and conclude that the utilization of enforcement results in a difference in the average energy consumption. Looking at the parameters that influenced this result the most, we see that the success of enforcement mechanisms was depended significantly on the population composition ($p < 0.0001$) which makes hypotheses 2–4 correct with respect to that parameter. Analysing Table 2 more closely, one notices that a high error rate can be observed, indicating that a difference exists between the three enforcement mechanisms that are grouped in the ANOVA. In order to examine this effect more closely, as well as to determine the extent to which each enforcement mechanism contributes to this difference, we perform Tukey's test as post-hoc analysis. Fig 2 shows the results of this analysis and Table 3 summarizes the statistical measures for each enforcement mechanism.

As the p-values in Table 3 show, all mechanisms have an average energy consumption significantly different to the experiments with no enforcement, however looking at the Tukey's test results in Fig. 2, it is clear that the results for the reputation mechanism stand out. Thus, whereas the results indicate that we can confirm hypothesis 1 for image-related information and police agents, Tukey's test for the reputation mechanism shows

[3] We performed the Shapiro-Wilk test and Levene's test to ensure the applicability of ANOVA. Due to limits on space we do not include details of these results.

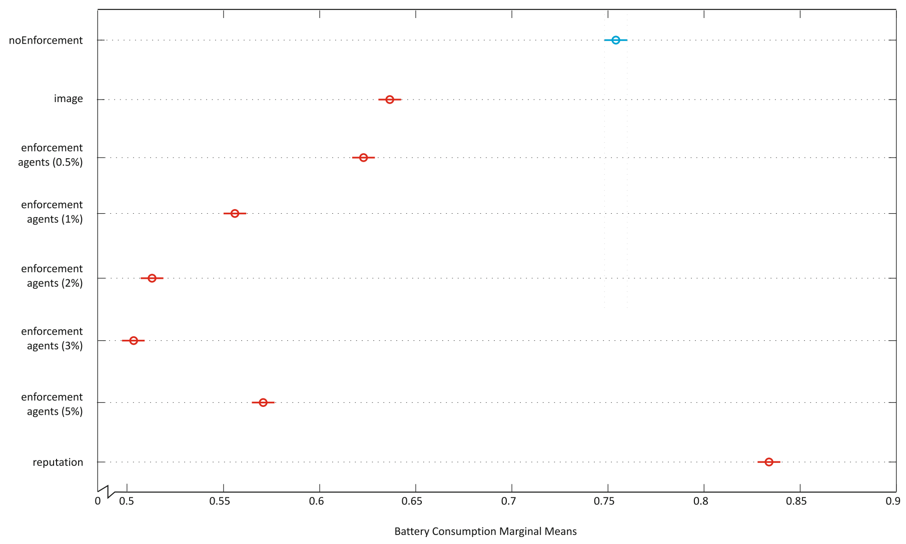


Fig. 2. Multiple Comparison (Tukey's Test) Results of Marginal Means for Comparing Simulation Experiments with and without Enforcement – Post Hoc Test

that the experiments using reputation, have an on average higher mean than when no enforcement is used. This implies that the utilization of reputation information *increases* the average energy consumption.

Testing Hypothesis 2. Digging deeper into the differences in energy consumption, the reason for this effect becomes apparent. As a result of the large number of messages arising from the transmission of reputation information, the communication costs are proportionately higher and thereby increase energy consumption. Thus, especially in cases with large numbers of honest agents, which would choose cooperation even without enforcement mechanisms, the reputation requests and answers do not improve enforcement, but rather result in greater energy consumption. This leads to reputation being worse with respect to the overall energy consumption ratio even compared to image information, i.e. settings where agents rely only on their own individual experiences.

This effect arising from additional communication can also be observed in the ANOVA and Tukey's test result for $\rho_{neighbourhood}$ (Figure 3), which show that a higher value of $\rho_{neighbourhood}$ tends to result in greater energy consumption[4]. This can again be attributed to the increased number of cooperation messages, because with a higher $\rho_{neighbourhood}$ more agents receive and send messages. These message costs outweigh the benefits of being able to observe more agents, because of the higher number of neighbours in the system. A second effect that impacts the reputation mechanism is the negative information introduced by malicious agents. Our simulation experiments

[4] The p-value for this relation is 0.3052, i.e. it is not significant. Nevertheless a tendency towards the described effect was detectable throughout the experiments.

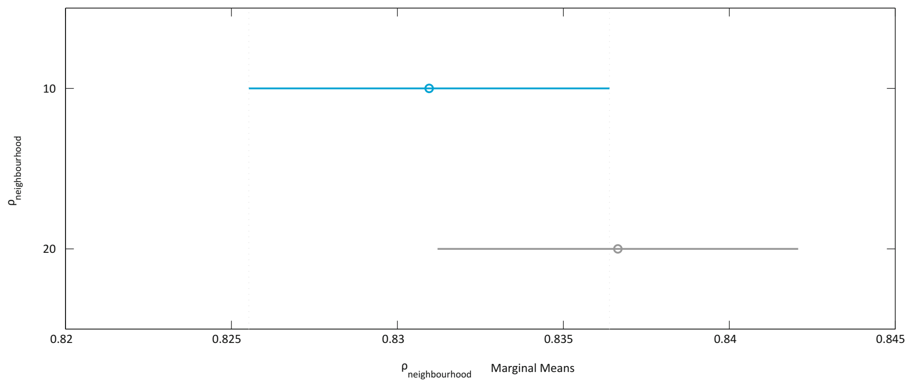


Fig. 3. Multiple Comparison (Tukey's Test) Results of Neighbourhood Density Marginal Means in Settings with Reputation Information – Post Hoc Test

were set up in such a way that if in doubt (i.e not verifiable), reputation information was considered to be correct (i.e. information from unknown sources was considered correct at the beginning). As a result of this, especially at the start of each experiment, by giving negative reputation information, malicious agents were able to discourage agents from cooperating with potential rivals, which also had the side-effect that agents could gather less image information of their own. As a consequence, they were more likely to have to trust uncertain reputation information in the following interactions, causing problems for the overall reputation mechanism. As a result of these findings, we conclude that the reputation mechanism chosen for our particular case study is unsuitable. The reasons for this however seem of a more general nature, that is, they are applicable to reputation mechanisms in general and should be considered when thinking about employing a reputation mechanisms for enforcement purposes: (i) transmission of reputation information incurs costs for both sender and receiver, which might result in reduced contribution of information, and the associated costs may outweigh the benefits (ii) in our experiments false information clearly harmed the system: a reputation mechanism therefore needs to be able to cope with false information, and (iii) most reputation mechanisms rely on small communities in order to function well: due to space limitations, we do not present the detailed figures, but our experiments indicate that an increase in the population size is deleterious for the reputation mechanisms, as more messages are sent and agents are less likely to encounter one another again soon.

Testing Hypothesis 3. A second interesting effect we can observe in Fig 2 is that an increase in the percentage of police agents does *not* result in a reduction of the average energy consumption ratio. Earlier on, in Table 3 we established that police agents can contribute to energy consumption reduction in WMGs, however exploring further, the value of this statement varies significantly with the percentage of police agents being employed (this is also indicated by a relatively high sum of squares error of 985.84 for the police agent value in Table 3). In the figure, the average mean energy consumption for experiments with 1%, 2% and 3% police agents are lower than the one with

5% and testing for significance, there is significant evidence against the null hypothesis that the means of the results with 1%, 2% and 3% police agents are not smaller than the results with 5% police agents (respective p-values < 0.0001). This implies that we have to reject the null hypothesis, which in turn means that the lower average energy consumption values for results with 1%, 2% and 3% police agents are not a result of chance. Comparing the number of defections in the experiments with police agents by performing a t-test, only a slight, and not significant, advantage for experiments with 5% police agents can be seen, i.e. in these settings police agents only added slightly to the total energy consumption. This implies that the improved detection of violations resulting from the larger number of police agents, is outweighed by the additional energy they consume. In economic terms this means that the lower percentage of police agents performs better with regard to satisficing cooperation when considering energy consumption. In economics, *satisficing* refers to a decision-making strategy that attempts to meet criteria for adequacy, rather than to identify an optimal solution [25]. Thus, although not optimal with regard to the detection of violations (1%, 2% and 3% police agents will detect less than 5%) the costs associated with them (i.e. the energy they consume for performing their observation and punishing actions) are significantly lower, making them more advantageous in terms of the overall energy saving.

Concerning the remaining parameters addressed in hypothesis 3, we found significant evidence that the null-hypotheses (i.e. that there is no impact on the parameters) can be rejected both for population composition and for neighbourhood density (respective p-values < 0.0001), while the significance levels for the size of the population vary between 0.0154 and 0.0604, which does not allow rejection of the null-hypothesis at a significance level of 0.01, but still indicates that there is reason to believe that population size has moderate impact.

Testing Hypothesis 4. For simulations in which image information is used, as hypothesized, we can reject the null hypothesis for all input parameters (i.e we have enough evidence to assume that hypothesis 4 is correct). Both the population composition and the population size have a significance level $p < 0.0001$ and $\rho_{neighbourhood}$ has a p-value < 0.0005.

Summarizing the Findings. Summarizing for the four hypotheses formulated in Sec. 5.1 we arrive at the following conclusions:

Hypothesis 1: Correct for police agents (between 1% and 5% of $| \mathcal{A} |$) and for image information, but incorrect for reputation information.

Hypothesis 2: Reputation information did not help to decrease the average energy consumption in the WMG. Both population size and composition had a significant impact here.

Hypothesis 3: The success of police agents is dependent on population density, composition and the number of police agents $| \mathcal{A}_{\mathrm{Enf}} |$.

Hypothesis 4: Correct for all parameters.

6 Related Work

Looking at previous works that are of importance for this paper, one can look into two different directions. The first direction is related work on means of enforcement in open distributed systems, such a WMGs, whereas the second direction is related work dealing with the comparison of enforcement mechanisms.

Looking at the first direction, one can identify a large literature in economics and social sciences on cooperation and free-riding and the mechanisms to overcome the latter. One of the most well-known analyses is by Ostrom [21], who shows that in small (relatively closed) communities these "tragedies" can be overcome. Other works use game theory [4] or evolutionary game theory [12,13] to address the question. In general, most of the works looking into enforcement use some form of punishment which serves as as a deterrent to the rational behaviour of utility-based participants (e.g. [9]). Thus, a punishment is a fine taken from the the participant's benefits. The topic has been framed mostly in terms of mechanism design and the issues that economists have studied more thoroughly are the information about infraction and sanctions [8], as well as the amount and pervasiveness of sanctions [6]. As pointed out before methodology often is either (evolutionary) game-theoretic (see [4] for example) or experimental (including agent-based simulations) [11,23].

In the second direction, i.e. the comparative study of enforcement, little related work can be found. In [5] for example, the authors discuss differences between image and reputation in detail, however no detailed experiments testing their impact on the same setting are made. Similar in [23] the authors tests the impact of how far messages are sent in a network and even considers the costs of these messages, but no comparison between trust and reputation is made.

7 Conclusions

We have compared three different enforcement mechanisms that are often employed in open distributed systems in the context of a single case study, namely WMGs. We use our analysis to examine the facilitation of cooperation are reduces the commons problem in WMGs. We show that by employing enforcement mechanisms we can reduce the cooperation dilemma inherent in WMGs and achieve overall positive effects with regard to a good that participants must contribute in order to sustain the WMG, namely battery power. WMGs broadly exhibit some of the most challenging characteristics of open systems, in the form of (potentially) rapidly changing participation and little or no authentication. Perhaps the most contentious aspect of the measurements, in respect of generalization, is the focus on energy consumption, since this is clearly specific to the WMG domain. The question is to what extent can this be viewed as a proxy for the fitness of an open system. We do not pretend that it is an accurate indicator, but being an aggregator of interactions within the open system, and its minimization being a metric for the effectiveness of the enforcement mechanisms, it seems likely to be positively correlated with the overall system fitness. A second issue affecting broader applicability may be the costs attributed to enforcement, since these are unavoidably expressed in domain-specific terms. Nevertheless, as with the previous point, the costs are just a formula associated with a transaction, and while changing the formula may lead to a different preference outcome, it should not fundamentally affect the validity of the approach.

Thus, the primary and unsurprising conclusion of this comparative analysis is that enforcement does not always help, and that the costs of enforcement need to be accounted for when deciding upon an enforcement mechanism, whether intrinsic in the form of, say communications, or extrinsic, in the form of, say rewards for observers looking for infractions. Of the three mechanisms presented, police agents especially seem to help to improve energy consumption. Although the results we present are inevitably linked to the specific mechanisms chosen, some general findings can be made.

The first is that the population composition accounts for the majority of the impact by the input factors on energy consumption. Second, the costs associated with the enforcement can outweigh the benefits. In the case of police agents this resulted in the situation that fewer agents produced a better absolute result in terms of energy consumption, while "only" satisficing the detection of violation actions. Similar effects could be seen in experiments with reputation information, where as a result of the particular reputation mechanisms chosen (i.e. Abdul-Rahman and Hailes) the message overhead produced by the reputation request and answers outweighed the benefits of the mechanism. One further aspect that influenced the performance of our reputation information-based enforcement mechanism negatively, is false information. This is particularly important as we implemented an adaptation of a mechanism that tried to account for this problem. However it did not have sufficient interactions to have any significant effect. The mechanism of Abdul-Rahman and Hailes – like any reputation mechanism – works better when the number of repeated interactions increases. Unfortunately this seems unlikely in a WMG and in open distributes systems in general, suggesting that the designer needs to consider – and possibly evaluate – carefully whether the characteristics of the situation are compatible with reputation mechanisms.

With respect to future work we aim to extend the simulation experiments by employing combinations of different enforcement mechanisms. In the experiments leading to this paper, we assumed that only one enforcement mechanism can be employed at any given time and that the choice of mechanism is constant throughout a simulation experiment. We plan to extend the work by relaxing this assumption and combining different enforcement concepts in the simulation experiments. This paper established the baseline for each mechanism in isolation.

The experiments could also be extended by employing more sophisticated agents. We employed three kinds of agents that pursue very different strategies in order to test how sensitive the simulation is to very one-sided behaviour (e.g. always defect or always cooperate). In an actual deployment of a WMG such a one-sided behaviour might not be very realistic. Therefore, agents with more sophisticated reasoning processes that exhibit more diverse responses to the successes or failures of cooperation situations are needed. One extension could be to allow malicious agents to cooperate occasionally in order to make them harder to detect for other agents, or to allow for variations in the reactions to sanctions by the utility maximizing agents. A further line of research might be to remove the assumption that police agents never cheat and repeat the simulation accounting for false information by police agents.

Acknowledgments. Tina Balke was partially supported by funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 288147.

References

1. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: HICSS (2000)
2. Balke, T., De Vos, M., Padget, J.A.: Analysing energy-incentivized cooperation in next generation mobile networks using normative frameworks and an agent-based simulation. Future Generation Computer Systems Journal 27(8), 1092–1102 (2011), `http://www.sciencedirect.com/science/article/pii/S0167739X11000574`
3. Balke, T., Villatoro, D.: Operationalization of the Sanctioning Process in Utilitarian Artificial Societies. In: Cranefield, S., van Riemsdijk, M.B., Vázquez-Salceda, J., Noriega, P. (eds.) COIN 2011. LNCS, vol. 7254, pp. 167–185. Springer, Heidelberg (2012), `http://dx.doi.org/10.1007/978-3-642-35545-5_10`
4. Coleman, J.S.: Foundations of social theory (August 1998)
5. Conte, R., Paolucci, M.: Reputation in Artificial Societies: Social Beliefs for Social Order. Springer (October 2002)
6. Dreber, A., Rand, D., Fudenberg, D., Nowak, M.: Winners don't punish. Nature 452, 348–351 (2008)
7. Esteva, M., Rodríguez-Aguilar, J.-A., Sierra, C., Garcia, P., Arcos, J.L.: On the Formal Specification of Electronic Institutions. In: Dignum, F., Sierra, C. (eds.) Agent Mediated Elec. Commerce. LNCS (LNAI), vol. 1991, pp. 126–147. Springer, Heidelberg (2001)
8. Fehr, E., Gächter, S.: Cooperation and punishment in public goods experiments. The American Economic Review 90(4), 980–994 (2000), `http://dx.doi.org/10.2307/117319`
9. Feldman, M., Papadimitriou, C., Chuang, J., Stoica, I.: Free-riding and whitewashing in peer-to-peer systems. In: Proceedings of the ACM SIGCOMM Workshop on Practice and Theory of Incentives in Networked Systems. ACM (2004)
10. Fitzek, F.H.P., Katz, M.D.: Cellular controlled peer to peer communications: Overview and potentials. In: Fitzek, F.H.P., Katz, M.D. (eds.) Cognitive Wireless Networks, pp. 31–59. Springer (2007)
11. Gurerk, O., Irlenbusch, B., Rockenbach, B.: The competitive advantage of sanctioning institutions. Science 312(5770), 108–111 (2006), `http://dx.doi.org/10.1126/science.1123633`
12. Güth, W., Ockenfels, A.: Evolutionary norm enforcement. Journal of Institutional and Theoretical Economics 156(2), 335–347 (2000), `http://edoc.hu-berlin.de/series/sfb-373-papers/1999-84/PDF/84.pdf`
13. Güth, W., Ockenfels, A.: The coevolution of trust and institutions in anonymous and non-anonymous communities. Discussion Papers on Strategic Interaction 2002-07. Max Planck Institute of Economics, Strategic Interaction Group (March 2002), `ftp://papers.mpiew-jena.mpg.de/esi/discussionpapers/2002-07.pdf`
14. Hardin, G.: The tragedy of the commons. Science 162, 1243–1248 (1968), `http://www.garretthardinsociety.org/articles/art_tragedy_of_the_commons.html`
15. Ionescu, M., Minsky, N., Nguyen, T.D.: Enforcement of Communal Policies for P2P Systems. In: De Nicola, R., Ferrari, G.-L., Meredith, G. (eds.) COORDINATION 2004. LNCS, vol. 2949, pp. 152–169. Springer, Heidelberg (2004)

16. Jones, A.J.I., Sergot, M.J.: A formal characterisation of institutionalised power. Logic Journal of the IGPL 4(3), 427–443 (1996), `http://www-lp.doc.ic.ac.uk/_lp/Sergot/InstitPower.ps.gz`
17. Kotz, D., Newport, C., Gray, R.S., Liu, J., Yuan, Y., Elliott, C.: Experimental evaluation of wireless simulation assumptions. In: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 78–82. ACM, New York (2004), `http://users.cis.fiu.edu/~liux/research/papers/axiom-mswim04.pdf`
18. Leibowitz, N., Ripeanu, M., Wierzbicki, A.: Deconstructing the kazaa network. In: Proceedings of the Third IEEE Workshop on Internet Applications. IEEE Computer Society (2003), `http://portal.acm.org/citation.cfm?id=832311.837393`
19. Miceli, M., Castelfranchi, C.: The role of evaluation in cognition and social interaction. In: Dautenhahn, K. (ed.) Human Cognition and Social Agent Technology. Benjamins, Amsterdam (2000)
20. Ostrom, E.: Governing the Commons: the Evolution of Institutions for Collective Action. Cambridge University Press (1990); 18th printing (2006)
21. Ostrom, E.: Coping with tragedies of the commons. Annual Review of Political Science 2, 493–535 (1999), `http://www.cipec.org/research/institutional_analysis/w98-24.pdf`, Workshop in Political Theory and Policy Analysis; Center for the Study of Institutions, Population, and Environmental Change, Indiana University, Bloomington, USA
22. Perrucci, G.P., Fitzek, F.H., Petersen, M.V.: Energy saving aspects for mobile device exploiting heterogeneous wireless networks. In: Heterogeneous Wireless Access Networks. Springer, US (2009)
23. Perreau de Pinninck Bas, A.: Techniques for Peer Enforcement in Multiagent Networks. Phd thesis, Universitat Autónoma de Barcelona (2010)
24. Sabater-Mir, J.: Trust and Reputation for agent societies. Ph.D. thesis, Institut d'Investigació en Intel.ligncia Artificial, IIIA (2003), `http://www.tesisenxarxa.net/TESIS_UAB/AVAILABLE/TDX-0123104-172828//jsm1de1.pdf`
25. Simon, H.A.: Rational choice and the structure of the environment. Psychological Review 63(2), 129–138 (1956)
26. Wrona, K., Mähönen, P.: Analytical model of cooperation in ad hoc networks. Telecommunication Systems 27(2-4), 347–369 (October 2004)

Shared Strategies in Artificial Agent Societies

Amineh Ghorbani[1], Huib Aldewereld[1], Virginia Dignum[1], and Pablo Noriega[2]

[1] Delft University of Technology, Faculty of Technology, Policy and Management, Delft, The Netherlands
{a.ghorbani,H.M.Aldewereld,M.V.Dignum}@tudelft.nl
[2] Artificial Intelligence Research Institute of the Spanish National Scientific Research Council, Barcelona, Spain
pablo@iiia.csic.es

Abstract. A shared strategy is a social concept that refers to a type of behavioural pattern that is followed by a significant number of individuals although it is, prima facie, not associated with an obligation or a prohibition. E. Ostrom has argued in favour of the pertinence of social strategies for institutional design and evolution and proposed a characterization suggestive of formal treatment. However, shared strategies as such have not been explicitly used in the context of regulated MAS in spite of their relevance and their affinity to more standard normative notions, of which a rich tradition exists in MAS research. In this paper, we discuss the notion of shared strategy, characterize its distinguishing features, propose its formalization using a temporal epistemic logic, and explore its potential use in regulated multi-agent systems.

1 Introduction

In the Netherlands, almost all people have dinner around 5:30pm. As a foreigner in that country, it is almost impossible to plan a (working) meeting around this time, which would be a ‘normal’ time in many other countries. On the other hand, having dinner that early is not an obligation. No one will be offended or would even care if you choose to eat later. In other words, there is no particular goal that everyone must reach following this strategy and therefore, individual disobedience would not have any particular consequence. One other important attribute of such statement is that it is more significant at the collective level rather than individual. In many cases people are not even aware of the common behaviour they are showing. Therefore, it is not a decision making action but rather more a routine-based reactive process. Nevertheless, knowledge of this typically Dutch behaviour, can help actors to plan their own, or joint activities (e.g., you can go to the supermarket at that time as it is likely to be very quiet, or you can invite your Dutch friends for dinner at that time). This is an example of a *shared strategy*, i.e. an institutional arrangement where different actors have the intention of performing the same task at a certain time or setting [17].

Even though the concept of shared strategy is socially and computationally very instrumental, it has not yet been implemented nor formalized in the MAS literature. First, it determines the general behaviour of the system thus providing expectations about the global behaviour of the system. For example, restaurants should start preparing meals early since there will be many people coming at that time. Second, this notion adds a new dimension to the deontic classical concept where there is no obligation, permission or prohibition, yet a shared behaviour takes place.

In MAS research, shared strategies can be a new way of expressing conventions that cannot easily be fitted into norms as they have no deontic 'flavour' to it. Shared strategies are different from collective intentions [7]. A collective intention is a goal shared by everyone in a team. Moreover, members of the team are aware of other agents intention to meet the common goal. For a shared strategy however, while all agents possibly have the same goal, their execution of tasks to fulfil the goal are independent of each other and if one agent does not perform the task, their goal can still be met. For example, while two people may have the collective intention to watch a new movie together that has just been released, many people share the strategy to watch movies as soon as they are released. In the former case, the whole objective of watching the movie will not be achieved if the two do not manage to watch the movie together, in the latter however, whether one watches the movie does not effect the general goal.

Regarding the benefits of implementing the concept of shared strategies in MAS, in this paper we take inspiration from the Institutional Analysis and Development framework (IAD), an institutional economic framework developed by the Nobel laureate Elinor Ostrom [18]. IAD is an analysis framework for understanding social systems with the purpose of (re)designing social rules (i.e. norms). The ADICO structure, part of the IAD framework, provides a language for institutional statements, such as shared strategies, institutional rules and norms [6].

The remainder of the this paper is structured as follows. In section 2 we explore the different definition of institutions and introduce the IAD framework and ADICO statements. In section 3 we further define shared strategies and formalize the concept. Section 4 discusses how this definition can be used in MAS. Section 5 explores related work. Finally, section 6 gives our conclusions and directions for future research.

2 Institutions

Institutional economics is an area of research in the social sciences with a rich collection of theories and frameworks that can be highly instrumental for MAS research. Among these is the Institutional Analysis and Development (IAD) framework which has gained popularity in different disciplines. A major focus of this framework is institutional statements defined as the ADICO sequences. We take inspiration from the ADICO definition to formalize shared strategy for artificial agent societies.

In institutional economics, institutions are defined as "the set of rules actually used by a set of individuals to organize repetitive activities that produce outcomes affecting those individuals and potentially affecting others" [18,15]. These rules include laws, regulations, social norms, and shared strategies amongst others. However, in MAS, the concept of institution usually refers not only to a set of rules as above but also to the regulative structures that enable verification and enforcement of those norms [4,1][1].

Institutions have two sides: on the one hand, they enable interactions, provide stability, certainty, and form the basis for trust. On the other hand, they may cause biased power relations. If institutions fail to fulfil stability or to enable decision making, there are grounds for institutional (re)design [13].

Institutional (re)design refers to the devising of new social arrangements, by examining existing arrangements and altering them when necessary [19]. I.e., institutional redesign refers to deliberate changes in institutional characteristics. In order to design institutions, one should be able to understand and analyse the institutional rules. Institutional frameworks such as the IAD framework by Ostrom [17] are developed for this purpose. This framework addresses the different components of a socio(-technical, -ecological) system that need to be analysed for institutional (re)design [17]. In the remainder of this section, we briefly introduce the IAD framework and the grammar of institutions (i.e. ADICO institutional statements).

2.1 Institutional Analysis and Design

The IAD decomposition of a social system is presented in figure 1. Its central concept is the 'action arena', in which individuals (or organizations) interact, exchange goods and services, engage in appropriation and provision activities, solve problems, or fight. The action arena is described by the participants (who have a set of resources, preferences, information, and selection criteria for action) and the action situation: the actual activity (or 'game') that is to be understood.

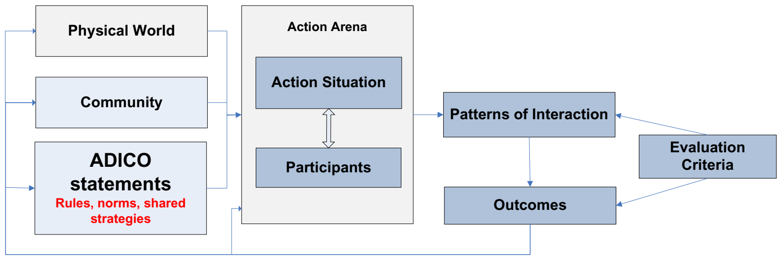


Fig. 1. The components of a social system in the IAD framework [18]

What happens in the action arena leads to patterns of interaction and outcomes that can be judged on the basis of evaluation criteria. The action arena itself is influenced by attributes of the physical world (e.g., climate, present technological artefacts), the attributes of the community in which the actors/actions

[1] Throughout this paper, we will be using the institutional economic terms where required.

are embedded (e.g. demographics), and the set of rules(referred to as institutional statements) that guide and govern the individuals behaviour.

Although physical world and community influence the action arena, it is the rules of the game or, in other words, the norms, that actually define it. Therefore, in IAD quite some attention is given to the institutional statements, which are decomposed into a structure (also referred to as grammar of institutions) called ADICO [17].

2.2 ADICO Institutional Statements

An ADICO statement consists of five components namely: *A*ttributes, *D*eontic, a*I*m, *C*ondition, and sanction (*O*r else). This decomposition is for the purpose of summarizing and analysing institutional statements[2], distinguishing between the different types and understanding the formation and evolution of these statements [17].

Attributes. Attributes describe the participants of an action situation to whom the institutional statement applies. Participants can be individuals who are distinguished by values such as age, sex or even roles in the system. For example, an attribute of an ADICO statement can be a 'student'. Corporate actors can also be considered as attributes instead of individuals (e.g., university). These actors can be distinguished by their organizational values such as location and size. The attribute component of an ADICO statement can never be empty. Therefore, if no attribute is specified for a given institutional statement the default value is 'all members' of the group.

Deontic Type. The purpose of the components is to distinguish between prescriptive and non-prescriptive statements. Deontic operators are *obligated* (O), *permitted* (P) and *forbidden* (H). While *obliged* and *forbidden* directly relate to the normative notions of 'ought', 'must' or 'should', *permitted* seems less related to the intuitive notion of norm. Permission rules however influence the structure of an action situation in three different ways. First by putting constraints on permissions and thus restricting actor behaviour. Second, assigning a permission to an action is constituting that action. Therefore, permission rules add action options to the action situation. Third, such rules grant rights to particular participants with certain properties to do an action. Some institutional statements don't have any deontic operator. As an example: "The person who places a phone call, calls back when the call gets disconnected".

Aim. The aim component describes the action or outcome (i.e., a state of affairs) to which the institutional statement applies. In order for a institutional statement to influence behavior, individuals must have a choice concerning its Aim. In other words, prescribing an action or outcome only makes sense if it its negation is also possible. E.g., the capability of voting implies the capability of not voting.

[2] We will use 'institutional statement' as a general term to address the concepts norm, rule and shared strategy.

Condition. Conditions are the set of parameters that define when and where an ADICO statement applies. If there is no condition stated it implies that the statement holds at all times.

Or Else. 'Or else' is the consequence of non-compliance to an assigned institutional statement. Only deontic statements include an 'Or Else'. A common type of 'Or else' is a sanction. Besides sanctions, rule violation may also result in the change of deontic (e.g. permitted to forbidden) of *another* rule. For example, it is forbidden to put a person in jail, but if they perform a crime, then the deontic changes to permission and one is allowed to imprison someone. Institutional actions may also be a result of norm violation. For example the role of the violator may be taken away. In general, the 'or else' component of an ADICO statement contains an institutional statement by itself which results in a nested structure of institutional statements. Also, the 'or else' component may be linked to the condition component that specifies the number of times that the norm has been violated.

According to the ADICO decomposition given above, an institutional statement can be divided into three different categories namely: rules, norms and shared strategies.

1. **ADICO**
   A *Rule*[3] (aka, regulatory rule) is the most complete form of statements covering all five components of the ADICO statement. In other words, rules have attributes, deontic type, action, condition and 'or else'.
2. **ADIC**
   A *Norm*[4] is an institutional statement without an 'or else' component. For example, shaking hands when being introduced to someone is a norm given that, if not done, it may affect your future relationship with that person. However, there is no fixed sanction and different people may have different reactions.
3. **AIC**
   A *Shared strategy* is an institutional statement where there are no sanctions or deontic type, and represents general expectations about the aggregate behaviour of others.

In the next section, we will discuss shared strategies in more detail.

3 Shared Strategies

3.1 Towards a Definition

According to E. Ostrom, a shared strategy is a social concept that refers to a type of behavioural pattern that is observed by a significant number of individuals although it is, prima facie, neither associated with any deontic modality,

[3] In agent literature, a rule is often addressed as 'norm' or 'regulation'.

[4] Sometimes called 'social norm' or even 'moral' or 'ethic code' in agent literature.

Table 1. Examples of Behaviours that can be assumed shared strategies

s_1	When a telephone conversation is cut, call back
s_2	When in Rome, do as Romans do
s_3	Dutch eat at 5:30
s_4	In a busy stairway, walk on the left
s_5	Jumping the queue is not nice
s_6	Faced with an unexpected obstacle, break
s_7	Only when a pedestrian makes a clear sign to attempt to cross the street, yield the right-of-way
s_8	If no police officer is in sight, skip the red light

nor having a reward or punishment linked to its performance. In order to elucidate the distinguishing features of shared strategies, in this section we explore different examples of social behaviour.[5]

Ostrom, in [17], pg. 143, proposes as an example of shared strategy, the rule of calling back when a telephone conversation is cut (s_1 in Table 1). Strategy s_1 is a conditional that under objective circumstances triggers an action. It does not explicitly entail an obligation or a prohibition, and no explicit or unique reward or punishment ensues. On a closer look, however, strategy s_1 may entail an *expectation*, that, depending on the context in which the interruption took place, may be a strong, possibly asymmetrical and, if not fulfilled may be consequential. The level and nature of expectation therefore reconciles with Ostrom's claim that, if an action rule is to be a shared strategy, then it would not matter whether α is done or not. We believe that the key is in the collective nature of expectations involved in shared strategies as we shall see.

Strategies s_2 and s_3 are similar to s_1 but their deontic component is more tenuous and thus closer to Ostrom's intuitive definition. Strategy s_2 "When in Rome, do as Romans do", like s_1, is an ostensible *directive for action* whose —relatively inconsequential— deontic component may guide the adaptive behaviour of foreigners, on one hand, and the leniency of natives towards non-standard behaviour of foreigners, on the other. Strategy s_3, "Dutch eat at 5:30", asserts a *factual regularity* but it also hides a directive for action whose compliance by an individual is indifferent to the rest of the world; nevertheless, under certain circumstances, it may have practical consequences (in Holland, for an individual's eating plans or for the operation of restaurants).

These three strategies may be deemed shared strategies only if we make some assumptions about the expectations involved explicit, otherwise they would be examples of *common* and *collective* strategies. Thus, strategy s_3 would be not a shared strategy but a "common strategy" if we understand it as a prevalent behaviour which people may not even be aware of. However, it becomes a "shared strategy" when we understand it as an expectation of common global behaviour;

[5] The concept of shared strategies has been addressed by social scientist using different terms. For instance, *scripts* by Schank and Abelson [21] or *conventions* by Hodgson and Knudsen [12]. For an overview of this literature see pg. 178 [17].

Table 2. Strategy Types

common strategy	most individuals do s_j
shared strategy	most individuals believe that most individuals do s_j
collective strategy	most individuals believe that most individuals believe most individuals do s_j

for instance, saying that most people believe that most Dutch eat at 5:30. Finally, s_1 also fails to be a shared strategy when the two parties expect that both parties should follow the rule, or technically, when there is collective belief. That is we have the three types of strategies characterized in Table 2:

Shared strategies may be situated, thus examples s_7 may only hold if in, say, Portugal. Furthermore, notice that some shared strategies (s_7 and s_8) may very well hold and be socially useful if situated in one context but may be highly dangerous patterns of behaviour in others, hence giving rise to full norms that forbid and punish their performance. Finally, situatedness is not only physical as s_5 "jumping the queue is not nice" illustrates. It is present in everyday situations like the supermarket or a theatre but becomes a strict directive (i.e., norm or rule) in surgery waiting lists and in some bureaucratic procedures.

As section 4 will show, it is important to distinguish between the collective character of a shared strategy –the fact that a collectivity has shared strategy or not— and whether each individual decides to enact or not that shared strategy in a particular moment. In fact, asymmetries of different types may create different expectations that affect agents' decisions; for instance, even when s_1 is a shared strategy, if I am calling a cab to go to the airport and communication breaks, it is me who should call back because it is in my best interest to continue the conversation and I may presume the cab doesn't know my number.

Likewise, shared strategies reveal a transient character that puts them between actual standard norms or social conventions, and fully unregulated behaviour, this transient character is revealed both in the collective and the individual perspectives. Thus, from an institutional perspective shared strategies can be seen as an *emerging social convention* or the *grounds for an emergent norm.* That is the case of s_4, "walking on the left of a busy stairway", that in London is a solid social convention —whose non-compliance is met with contempt or derision, while in Paris it is a shared strategy, and in the US it is not (still?). Note also that driving on one of either sides of a road, which was a shared strategy at some point, became institutionalized as an explicit norm everywhere; probably because of the social significance of non-compliance. From an individual's perspective, on the other hand, the transient character of shared strategies is evident in the same strategy s_4 that may be likened either to an *internalized norm* or to a *tacit social convention* of which the subject might be not fully aware.

3.2 Formalizing ADICO Statements

In this section we formalize the notion of *Institutional Statement* from Ostrom [17] to get to a semantic description of the rules, norms and, foremost, shared strategies of the ADICO framework. This forms the basis of our discussion on shared strategies in the next section.

The logic used for the formalization is a temporal epistemic logic based on CTL* [8] for the temporal aspect and KD45 [14] for the epistemic aspect. We use a technique similar to [9] for the combination of these modalities. In short, the resulting logic is a temporal logic where the states contain an epistemic modality. This allows for the expression of beliefs and changes of beliefs, but not the expression of beliefs about the temporal structures (that is, one can change its beliefs in a future state, but one cannot have beliefs about future or past states).

The core of the logic is given by the set of propositions $\mathcal{P}$, which can be used to construct sentences using the typical propositional operators ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$). The set of all possible well-formed propositional formulas will be denoted as $\mathcal{L}_{\mathcal{P}}$. This logical core is extended to an epistemic logic of beliefs using a belief-operator (B), following the KD45 principles, resulting in a set of well-formed sentences $\mathcal{L}_{\mathcal{BP}}$. The temporal logical language $\mathcal{L}_{\mathcal{TBP}}$ is then constructed by adding the usual temporal operators: path operators A (all paths), E (some paths), and state operators X (next), G (always), F (sometime), U (until). The language is further enriched with *stit*: e_r ('see to it that', see [5]) to express individual action.

Using the logic $\mathcal{L}_{\mathcal{TBP}}$ we can now introduce the syntax of ADICO institutional statements as follows.

Definition 1 (Institutional Statement). *ADICO Institutional Statements are of the form*

$$D_R(I \mid C) \rightsquigarrow o$$

where
- D represents one of the modalities: $\{O, P, H, S\}$
- R being the attribute*, represented as a set of* roles*;*
- I being the aim*, represented as an expression from $\mathcal{L}_P$;*
- C being the condition*, represented as an expression from $\mathcal{L}_P$; and*
- $\rightsquigarrow o$ being the or-else*, where o is represented as combination of institutional statements.*

The modality of an institutional statement can either be: O (obligation), P (permission), H (prohibition), or S (shared strategy). The modality determines the semantics of the statement. Roles in our framework are considered as labels, with $\mathcal{R}$ being the set of all roles in the institution. The applicability of an institutional statement is thus $R \subseteq \mathcal{R}$. The $\rightsquigarrow o$ part of the statement expresses the *or-else* of the institutional statement, representing the reaction to violations of the statement. Intuitively, this means that when the lefthand-side of the $\rightsquigarrow$-operator is violated, the righthand-side of the $\rightsquigarrow$-operator is activated. The reaction, o, is represented as an expression containing institutional statements combined with conjunctions and disjunctions. It is also possible that $o \equiv \top$, which expresses that the institutional statement has no reaction[6].

The different types of institutional statements referred to by Ostrom can be obtained in the following ways. A *rule* is an institutional statement that contains all elements, and where the modality is of deontic nature (that is, $D \in \{O, P, H\}$). *Norms* are institutional statements with a deontic modality ($D \in \{O, P, H\}$) and where no o is specified; $D_R(I \mid C)$. Finally, *shared strategies* are institutional statements without a deontic modality ($D = S$) and where the reaction o is absent; $S_R(I \mid C)$.

For the semantics of the institutional statements, we create reductions of the newly introduced operators to the basics of the $\mathcal{L}_{\mathcal{TBP}}$. Due to space limitations, we give the reduction of obligations, prohibitions and shared strategies; the reduction of permissions (weak permissions, strong permissions, cf. [23]) is out of scope of this paper, and left as an exercise to the reader.

Definition 2 (Reduction of Obligations).

$$\begin{aligned} O_R(I \mid C) \rightsquigarrow o \Leftrightarrow \forall r \in R A\Big[& C \to (\neg viol(I, r))\ U \\ & \big(e_r\, I \wedge X(AF \neg viol(I, r)) \vee \\ & X(\neg I \wedge viol(I, r))\big) \\ \wedge\, & viol(I, r) \to o\Big] \end{aligned}$$

The above definition transforms the obligation into a $\mathcal{L}_{\mathcal{TBP}}$ sentence, using an Anderson's reduction [2], similarly as done in, e.g., [1]. Intuitively, the definition expresses that whenever the condition (C) holds, *either* the aim (I) is achieved by those obliged ($e_r\, I$), in which case no violation of the obligation will ever occur, *or* the aim is not achieved, and a violation happens. Moreover, when the violation happens, the reaction statement o (if present) is triggered (these statements typically express sanctioning mechanisms, see [17]).

Definition 3 (Reduction of Prohibitions)

$$H_R(I \mid C) \rightsquigarrow o \Leftrightarrow O_R(\neg I \mid C) \rightsquigarrow o$$

The reduction of prohibitions is based on the principle that $Hp \equiv O\neg p$ from most deontic logics.

Definition 4 (Reduction of Shared Strategy)

$$S_R(I \mid C) \Leftrightarrow \forall r_1 \in R, \forall r_2 \in R \backslash \{r_1\} : A(C \to B_{r_1} e_{r_2} I)$$

The reduction of shared strategies is formed around the idea that shared strategies represent an *expectation.* Intuitively, a shared strategy expresses the expectation that other members of the same group (i.e., playing the same role, or

[6] Typically, when $o \equiv \top$, we omit the $\rightsquigarrow o$ part of an institutional statement for readability: $D_R(I \mid C) \rightsquigarrow \top = D_R(I \mid C)$.

part of the group of roles that share the strategy) will try to follow the shared strategy. This idea is reflected in definition 4. This is different from the notions of common strategy, where everyone in the group does the expected thing, and joint strategies, where everyone in the group intends that they do the expected thing. Using similar elements as used in definition 4, we can also formalize the notions of common strategy and joint strategy:

Proposition 1 (Common & Joint Strategies)

$$\begin{aligned} CS_R(I \mid C) &\Leftrightarrow A(C \rightarrow \forall r_1 \in R : e_{r_1} I) \\ JS_R(I \mid C) &\Leftrightarrow A(C \rightarrow \forall r_1, r_2 \in R : B_{r_1} B_{r_2} e_{r_1} I) \end{aligned}$$

Common strategies (CS) happen when all agents in a system are programmed alike, and act in similar manners; that is, every member of a group R follows a common strategy CS_R to do I when each member of that group does I. A joint strategy (JS), similar to joint-intentions [7], is when every member of a group R does I, but also knows (and expects) that every other member of R also does I. That is, there is shared belief that the group beliefs that they are doing I.

By formalizing the shared strategies (and similarly, common and joint strategies) we lost an aspect of Ostrom's concept. An important aspect of Ostrom's reading is that a shared strategy can be not acted upon, which is missing from definition 4, since we expect that every agent in the group will do I. Informally, definition 4 reads as "everyone from group R believes everyone from group R does I". Ostrom's reading of a shared strategy is more in line with "most from group R believe that most of group R do I" (see the discussion earlier in section 3.1). This has an impact on the way agents behave, because in the first reading one can be sure that members of the group R will do I, whereas in the second reading it might be that some members of R will not do I. Therefore, we need to weaken our definition, for which we require a semantic definition of 'most'.

Definition 5 (Most). *We define the set-theoretic 'most' operator* W *as follows, for a set of roles R:*

$$\mathsf{W}(R) = R' \Leftrightarrow R' \subseteq R \wedge (|R'| > 1/2 \cdot |R|)$$

Intuitively, this definition expresses what one would expect. If R' is representing the most of set R, then at least half of the agents in R are also in R'; that is, R' is a subset of R and the number of elements of R' is at least half that of R.

Using the concept of 'most' we can create weaker versions of the earlier strategies as follows.

Proposition 2 (Weak strategies)

$$\begin{aligned} CS^-_R(I \mid C) &\Leftrightarrow A(C \rightarrow \forall r_1 \in \mathsf{W}(R) : e_{r_1} I) \\ JS^-_R(I \mid C) &\Leftrightarrow A(C \rightarrow \forall r_1, r_2 \in \mathsf{W}(R) : B_{r_1} B_{r_2} e_{r_1} I) \\ S^-_R(I \mid C) &\Leftrightarrow \\ &\qquad A(C \rightarrow \forall r_1 \in \mathsf{W}(R), \forall r_2 \in \mathsf{W}(R \backslash \{r_1\}) : B_{r_1} e_{r_2} I) \end{aligned}$$

Table 3. Examples of Shared Strategies

s_1	$S^{-}_{on\_phone}(call\_back \mid conversation\_cut)$
s_2	$S^{-}_{tourist}(do\_as\_Roman \mid in\_Rome)$
s_3	$S^{-}_{Dutch}(eat \mid 5:30)$
s_4	$S^{-}_{pedestrian}(stay\_left \mid in\_busy\_stairway)$
s_5	$S^{-}_{civilised\_people}(\neg jump\_queue)$
s_8	$S^{-}_{driver}(skip\_red\_light \mid \neg police\_in\_sight)$

The expressions in proposition 2 represent the weakened versions of the expressions in proposition 1 and definition 4. Intuitively, they read as follows. A group R has a *weak common strategy* to I when most of R do I. A group R has a *weak joint strategy* to I when most members of R believe that most other members of R believe that most of them do I. Finally, a group R has a *weak shared strategy* to I when most members of group R believe that most other members of R do I.

A formalization of some of the examples from table 1 is shown in table 3 below.

4 Shared Strategies Applied in MAS

In this section, we discuss the practical application of shared strategies in MAS. Shared strategies can be seen as a form of regulation of individual behaviour within a system, or as mechanisms to improve cooperation, coordination and control in MAS. As such, shared strategies can be used by agents in their reasoning processes, in order to determine their plans in a shared environment (cf. section 4.1), or as means to support design and evaluation of engineered MAS (cf. section 4.2).

4.1 Individual Application

In this section, we look at how shared strategies can be used by individual agents in their planning. As with norms, agents can and should take into account the shared strategies holding in a domain in order to generate efficient plans for their goals. We assume here autonomous cognitive agents that are able to use their knowledge about a domain in the generation of plans. Such agents can decide on the adherence or not to norms. Other researchers have studied norm-based planning [22], i.e. the generation of optimal plans with respect to a set of norms. In this section, we concentrate on the use of shared strategies for the generation of plans.

The intuition of the formal definition of shared strategy introduced in section 3 is that most agents assume that under certain conditions, other agents will behave in a certain way. While common strategies may be designed into agent systems so that agents are not aware of them as common behaviour, shared strategies can be perceived by the agents as shared behavioural patterns. If most agents see that most agents have this new perception, the strategies will be globally recognized as shared strategies. This new knowledge will then be updated in their belief system and used in their planning. Based on these new beliefs, agents can take two approaches to use shared strategies in their planning, referred here as an *optimistic* and a *pessimistic* approach. In order to discuss the difference between these two approaches, we take as example the shared strategy:

$$S^{-}_{drivers}(break|obstacle\_in\_road)$$

which represents the fact that drivers will break when there is an obstacle in the road.

An optimistic pedestrian agent will assume that all drivers will break when she crosses the road, and therefore will plan to cross the road even if she sees a car approaching. On the other hand, a pessimistic pedestrian will assume that you cannot know which drivers will adhere to the shared strategy, since not all have to follow it, and therefore will plan to stop at the curb when she sees a car approaching.

We are currently working on an extension to the BDI architecture that incorporates reasoning using shared strategies.

4.2 Institutional Application

Form an institutional perspective there are two issues worth identifying. The relationships between shared strategies and institutional design and evolution, and the role of shared strategies in multiagent-based simulation.

Since shared strategies constitute a regularity of the aggregate behavior, institutional conventions may be designed to promote or to control the consequences of that regularity. The approach is straightforward when the existence of a shared strategy is known in advance and it is likely that its execution carries out institutional objectives. In this case, it is reasonable to include specific evaluation mechanisms to monitor the effects of the strategy, and use these to assess transaction costs that would in turn guide the adaptation of the institution to actual performance ([14]). Concomitantly, it is also feasible to establish institutional norms and conventions —with the appropriate evaluation mechanisms— that regiment, constrain or foster the enactment of the shared strategy by participating agents.

The way of dealing with the alternative case is less obvious. When the existence of a shared strategy is not known in advance, ordinary performance monitoring does not necessarily identify the behavioral regularity, even when performance indicators might signal a hidden cost. In such case, institutional reaction may be untimely and inefectual. To contend with such eventuality, one may attempt to foresee undesirable outcomes and, at the risk of overregulation, legislate against them. The opacity of undesirable outcomes, however, may sometimes be appropriately addressed with conventional mechanism-design techniques or by a clever use of modeling and simulation methodologies.

In addition to their value for visualizing the effect of shared strategies on institutional performance, in this context, the modeler deals with the system as a regulated MAS, making a shared strategy a feature of individual agents and harnessing individual actions through institutional conventions of different sorts. The use of shared strategies may be fruitful for some forms of agent-based simulation. One relevant form is to use shared strategies as a salient part of the agents' internal decision models. This way, the designer may study different aspects of normative, motivational and goal-directed attitudes (for example the interplay of norms and strategies in different agent architectures, norm internalization processes, norm emergence, norm compliance vs. conflict resolution approaches, value formation, achievement degrees). Another form of using shared strategies in agent-based simulation is to factor the analysis of aggregate behavior by designing populations partitioned by shared strategies, thus measuring cost and value of interactions within populations with pure and mixed strategies, rational or spontaneous triggering of the shared strategies, etc.

4.3 Institutional Emergence

Although this is not the primary focus of this paper, we see the ADICO structure as an instrumental tool to study the emergence of rules, norms and shared strategy in agent societies.

As Ostrom explains in [17], the change in any part of the ADICO statement results in the evolution of such entities in a society. For example, when global expectations about a shared strategy narrow down to individuals, a deontic flavour emerges, turning the shared strategy to a norm. Likewise, when the implicit, non-unique and unclear consequences of non-compliance to a norm become common, known and explicit to everyone, that norm turns into a sanction.

Besides the study of institutional evolution as we described above, the ADICO statement can be linked to the internal architecture of an agent (e.g. BDI) so that the agents can perceive common behaviour and recognize and establish it as an institutional statement. For example, if the agent detects the components of an ADICO structure in a repeated pattern of behaviour in the society it will announce this as a shared strategy/rule/norm, and if many agents announce the same statement, this will become an emergent ADICO statement in the artificial society. We are only addressing the idea of this application. However, the implementation will be the topic for future work.

5 Related Work

Some concepts in the MAS literature address shared strategy to some extent. Table 4 shows some of the most relevant concepts and compares their usage with similar examples. Normative information can be situated in the environment (e.g. sign boards) which means that a norm only needs to be followed within a certain boundary of space and time [16]. The type of situated norm can be warning, obligation and direction. A shared strategy however, does not necessarily have to be bound to location and time or have any of the types given to distributed norms (i.e. warning, obligation, direction).

Social conventions are rules that restrict agent behavior while having no threat or punishment. Young (1993) presents the following definition of a conventional norm: "A convention is a pattern of behavior that is customary, expected, and self-enforcing. Everyone conforms, everyone expects others to conform, and everyone wants to conform given that everyone else conforms."

For a shared strategy however, no one has expectation for others to conform because they are not aware if the person is necessarily a follower of the strategy. No (low) expectation results in no (low) disappointment. For example, if in a given context calling back if the line is dropped is a social convention, then the person may be upset but if it is a shared strategy, the person does not know if he is a performer of the shared strategy 'calling back', and thus will not be offended if the caller does not call back. Therefore it can also be concluded that a shared strategy has lower priority than a convention for agent planning.

A collective intention is the reason for team existence and it implies that all members intend for *all* others to follow that intention [7] . The goal of the team may not be reached if one agent may decide not to follow the intention. However, for a shared strategy, as mentioned previously, most people know the strategy and know that most others will follow the strategy. Therefore, there is no obligation for agents to perform the strategy and there is also no significant consequence on an individual level while the global behavior of the system may be important.

Table 4. Concepts related to shared strategy in current MAS literature

Concept name	ref.	Example
Shared Strategy	[17]	The Dutch eat dinner at 5:30pm.
Situated Norm	[16]	In this ship dinner is served at 5:30 pm (or else no food).
Social Norms/ Conventions	[24]	When eating dinner, people start at the same time
Shared/Collaborative plans	[11,10]	Those group of friend have plan to make dinner together
Collective Intention	[7]	Those group of friend are committed to have dinner together at 5:30 pm.

Norm internalization [3] is another topic of research in MAS that can be used in combination with shared strategies. Norm internalization is progressive. This is in line with the transition of ADICO statement from one type to another (e.g. a norm becomes a shared strategy)[16, 3]. In other words, during the process of internalization, an ADICO rule which has all five parts of the statement, may loose the 'or else' and become a norm and later on turn into a 'shared strategy' by losing the deontic. On the other hand, the more the norm is internalized the less decision making is required. This again is in line with the definition of shared strategy which is more of a routine that requires less thinking. A fully internalized norm is a shared strategy only if it is shared among people.

The original formulation of shared plans [10] does not see the necessity for an agent to have intentions towards the act of another agent. It is similar to shared strategies in the sense that there is not joint intention between the agents. However, it is different to shared strategies because the agents make plans and actually coordinate in performing the action. Collaborative plans [11] which are a revised version of shared plans are also different from shared strategies because they produce commitment to the joint activity.

6 Conclusion and Future Work

In this paper we presented the concept of shared strategy as an alternative concept to that of norm in MAS. Based on the work of Ostrom, namely the notion of ADICO institutional statement, we presented an integrated formalism to describe the semantics of norms and shared strategies, based on a temporal epistemic logic.

A shared strategy is a low priority statement leading to action among a group of agents. Since the expectation is *shared*, each agent believes that *most* other agents will perform the action but does not necessarily know who. Therefore, agents don't have expectations for a particular other agent to perform shared strategies because they cannot know whether that particular agent follows the strategy or not, even though as a group, most will. This yields that no deontic type and no sanction can be assigned to a shared strategy.

Shared strategies are a crucial part of agent societies as they result in global behaviors that may need to be taken into consideration by other agents who may be part of the system or merely global viewers. A shared strategy can change into norm and vice versa depending on the level of norm internalization and the context which facilitates the implementation of norm emergence and evolution [20].

For future work, we are further extending the formalization of shared strategy. We are also exploring how shared strategies can be implemented into BDI architecture.

Acknowledgments. This research was partly funded by the COST Action IC0801 "Agreement Technologies" Grant.

References

1. Aldewereld, H.: Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols. SIKS Dissertation Series 2007-10. Utrecht University, PhD Thesis (2007)
2. Anderson, A.: A reduction of deontic logic to alethic modal logic. Mind 67, 100–103 (1958)
3. Andrighetto, G., Villatoro, D., Conte, R.: Norm internalization in artificial societies. AI Communications 23(4), 325–339 (2010)
4. Arcos, J., Esteva, M., Noriega, P., Rodrguez, J., Sierra, C.: Engineering open environments with electronic institutions. Journal on Engineering Applications of Artificial Intelligence 18(2), 191–204 (2005)

5. Belnap, N., Perloff, M.: Seeing to it that: a canonical form for agentives. Theoria 54(3), 175–199 (1988)
6. Crawford, S., Ostrom, E.: A grammar of institutions. American Political Science Review, 582–600 (1995)
7. Dunin-Keplicz, B., Verbrugge, R.: Collective intentions. Fundamenta Informaticae 51(3), 271–295 (2002)
8. Emerson, E.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 955–1072. MIT Press (1990)
9. Engelfriet, J.: Minimal temporal epistemic logic. Notre Dame Journal of Formal Logic 37(2), 233–259 (1996)
10. Grosz, B., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence 86(2), 269–357 (1996)
11. Grosz, B., Sidner, C.: Plans for discourse. Technical report, DTIC Document (1988)
12. Hodgson, G., Knudsen, T.: The complex evolution of a simple traffic convention: the functions and implications of habit. Journal of Economic Behavior & Organization 54(1), 19–47 (2004)
13. Klijn, E., Koppenjan, J.: Institutional design. Public Management Review 8(1), 141–160 (2006)
14. Meyer, J.-J.C., van der Hoek, W.: Epistemic Logic for AI and Computer Science. Cambridge University Press (1995)
15. North, D.: Institutions, institutional change and economic performance. Cambridge University Press (2009)
16. Okuyama, F., Bordini, R., da Rocha Costa, A.: Spatially distributed normative objects. In: Coordination, Organizations, Institutions, and Norms in Agent Systems II, pp. 133–146 (2007)
17. Ostrom, E.: Understanding institutional diversity. Princeton Univ. Pr. (2005)
18. Ostrom, E., Gardner, R., Walker, J.: Rules, games, and common-pool resources. Univ. of Michigan Pr. (1994)
19. Pettit, P.: Institutional Design and Rational Choice, pp. 54–89. Cambridge University Press (1996)
20. Savarimuthu, B., Cranefield, S., Purvis, M., Purvis, M.: Norm emergence in agent societies formed by dynamically changing networks. Web Intelligence and Agent Systems 7(3), 223–232 (2009)
21. Schank, R., Abelson, R., et al.: Scripts, plans, goals and understanding: An inquiry into human knowledge structures, vol. 2. Lawrence Erlbaum Associates, Nueva Jersey (1977)
22. Panagiotidi, S., Vázquez-Salceda, J.: Normative Planning: Semantics and Implementation. In: 13th International Workshop on Coordination, Organizations, Institutions and Norms in Agent Systems (COIN@WI-IAT), Lyon, France (2011)
23. van der Torre, L.: Deontic Redundancy: A Fundamental Challenge for Deontic Logic. In: Governatori, G., Sartor, G. (eds.) DEON 2010. LNCS, vol. 6181, pp. 11–32. Springer, Heidelberg (2010)
24. Villatoro, D., Sen, S., Sabater-Mir, J.: Of social norms and sanctioning: A game theoretical overview. International Journal of Agent Technologies and Systems (IJATS) 2(1), 1–15 (2010)

Goal-Directed Policy Conflict Detection and Prioritisation

Mukta S. Aphale*, Timothy J. Norman, and Murat Şensoy

Dept. of Computing Science,
University of Aberdeen,
Aberdeen, UK
{m.aphale,t.j.norman,m.sensoy}@abdn.ac.uk

Abstract. A policy (or norm) is a guideline stating what is allowed, forbidden or obligated for an entity, in a certain situation, so that acceptable outcomes are achieved. Policies occur in many types of scenarios, whether they are loose social networks of individuals or highly structured institutions. It is important, however, for policies to be consistent and to support their goals. This requires a thorough understanding of the implications of introducing specific policies and how they interact. It is difficult, even for experts, to write consistent, unambiguous and accurate policies, and conflicts are practically unavoidable. In this paper we address this challenge of providing automated support for identifying and resolving logical and functional conflicts.

Keywords: Policies, Norms, Policy Authoring, Conflict Resolution, Intelligent Agents.

1 Introduction

Policies are designed to guide and regulate behaviour of entities in a system; they are system-level constraints that represent ideals of behaviour. The benefits of a policy-based approach include reusability, extensibility, context-sensitivity, verifiability, information security, support for simple and sophisticated components and reasoning about component behavior [13].

Policies can be classified into two categories — collective and individual. Collective policies are sets of rules applicable to entities in a particular group or entities playing a specific role. They represent an agreement among agents who are responsible for defining the rules and ensuring that common goals are promoted. For example, the NHS Care Record Guarantee for England is an agreement between NHS (National Health Service) and patients, brokered by patient organisations and senior healthcare experts. Personal preferences of entities are

* The research described here is supported by the award made by the RCUK Digital Economy programme to the dot.rural Digital Economy Hub; award reference: EP/G066051/1.

also often expressed by individual policies. For example, rules for sharing personal information in a social network. Policies operate in conjunction with individual and organisational goals, such as the provision of effective patient care. The key challenge here is: how to specify policies/norms that protect important information (secrets) and promote ideal action (normal operating procedures), while ensuring the achievement of individual/organisational goals?

It is difficult, even for experts to write consistent, unambiguous and accurate policies. Policies may conflict with each other and with organisational goals in many situations; i.e., logical and functional conflicts, respectively [5]. Identifying and resolving such conflicts, and functional conflicts in particular, is an important challenge. This question has been explored in the context of practical reasoning by Kollingbaum [9], where an agent architecture (NoA) is proposed that detects direct and indirect conflicts between norms and action choices. Conflicts are then either resolved through heuristic strategies or labelled for further deliberation. Conflicts between beliefs, obligations, intentions and desires are also explored in the BOID architecture [4], where the aim is to identify maximal subsets of consistent norms and intentions. Reasoning about plans and norms has been further addressed by Meneguzzi [10], where Jason [2] was extended to include normative constraint checking. In this way, agents can modify their behaviour in response to newly accepted norms, by creating new plans to comply with obligations and suppressing the execution of existing plans that violate prohibitions.

In other research it has been demonstrated how intelligent agents can assist humans in complex, norm-governed decision making [12] and prognostically reason about possible normative violations and replan to avoid these violations [11]. In addition to supporting human decision-making constrained by norms or policies, existing research has addressed the problem of supporting humans in authoring policies. In this area, however, there is very limited automated support for conflict detection and resolution. The authoring process is guided through the use of templates in the work of Johnson et al. [8]. In Uszok et al.[14] some reasoning support for conflict detection has been explored, but this is confined to the detection and resolution of logical conflicts.

In this research, we focus on the support of policy authors in conflict detection and resolution including functional conflict detection. Conflicts occurring in a system may be of varying significance. More critical are the those that have higher chances of occurring and those that can impair goal achievement. Hence, it is crucial to identify conflicts, the resolution of which will lead to maximum likelihood of goal achievement. The key questions addressed are: how to identify conflicts that are most relevant given the domain and goals of organisation, and how to resolve conflicts that are most important.

This paper is organised as follows: In Section 2 we introduce the Polar Agent application developed for policy authoring and planning. Our notions of policies and conflicts are detailed in Section 3. Our activity prioritisation model for ranking the activities within a set of plans to achieve a goal is described in Section 4. Conflict detection methods implemented by the Polar Agent are described in

Section 5, where specific emphasis is given to functional conflict detection. The conflict resolution method implemented by the Polar Agent and scope for future work are outlined in Section 6, and in Section 7 we present our conclusions.

2 Polar Agent

In support of this research, we have developed the Polar Agent for authoring OWL-POLAR (OWL-based POlicy Language for Agent Reasoning) [6] policies and reasoning about interactions between plans and such policies (see Section 3). It consists of two core modules — Policy Authoring and Domain Analysis. Conflict detection and resolution mechanisms are implemented by the authoring module, and possible plans for a specified goal are generated and analysed using the domain analysis module. An intelligent agent assists the user in understanding the implications of policies and any conflicts that may occur between policies and between policies and plans. The architecture of the Polar Agent is illustrated in Fig. 1.

1. ***Policy Authoring.*** This consists of an interface for specifying policies using OWL-POLAR and machinery for the detection and resolution of conflicts.
   - ***Conflict detection.*** Potential logical and functional conflicts are assessed according to the relative priorities of activities. Priorities are computed on the basis of the cost of achieving goals and the probability of successful plan execution.
   - ***Conflict resolution.*** Our aim is to support users in the resolution of conflicts, and so the focus here is on the presentation of options to refine policies in order to resolve conflicts. The intelligent agent assistant

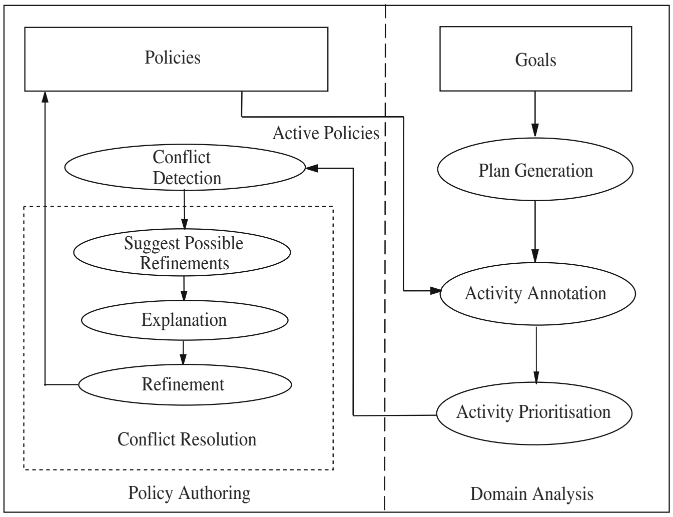


Fig. 1. Polar Agent Architecture

provides recommendations regarding how conflicts my be resolved and the implications of policy refinements.

2. ***Domain Analysis*** contains an interface for specifying goals and mechanisms for the automatic generation and analysis of possible plans. There are three core processes involved here:
   – ***Plan Generation*** generates all possible plans for a particular goal.
   – ***Activity Annotation*** labels activities in possible plans with policies that *may* constrain these activities, the cost of achieving goal, etc.
   – ***Activity prioritisation*** prioritises activities on the basis of factors including cost, frequency of occurrence in plans and probability of success.

In this paper, our focus is on the prioritisation of conflict detection, and resolution driven by an organisation's goals. Before presenting our model of prioritised conflict detection, however, we must clarify precisely what we mean by policies and conflicts.

3 Policies and Conflicts

OWL-POLAR, developed in prior research [6], is used as a language for expressing policies in this work. OWL-POLAR is sufficiently expressive to be used for specifying policies in real-life applications. It is based on OWL-DL and exploits OWL 2.0 that includes a sophisticated set of built-in numeric data ranges and expressive constructors for building new data ranges. The expressiveness of OWL-POLAR is not restricted to DL — use of variables is allowed while defining policies by means of conjunctive semantic formulae and data ranges are exploited to enable the expression of complex constraints on policies. Means for explicitly defining expiration conditions for policies are provided in OWL-POLAR.

Advanced policy analysis that is not limited to subsumption checking is implemented in this framework. For example consider following policies: *(i) Dogs are prohibited to enter in a restaurant*, and *(ii) A member of CSI team is permitted to enter a crime scene.* There is no subsumption relationship between these policies. However, the OWL-POLAR reasoner anticipates a conflict by composing a state of the world where these policies are in conflict; e.g. the crime scene is a restaurant and there is a dog in the CSI team.

An OWL-DL ontology $o = (TBox_o, ABox_o)$ consists of a set of axioms defining the classes and relations ($TBox_o$) as well as a set of assertions about individuals in the domain ($ABox_o$). Concept axioms are of the form $C \sqsubseteq D$, where C and D are concept descriptions. Relation axioms are expressions of the form $R \sqsubseteq S$, where R and S are relation descriptions. The ABox contains concept assertions of the form $C(a)$, where C is a concept and a is an individual name; and relation assertions of the form $R(a, b)$, where R is a relation and a and b are individual names.

Conjunctive semantic formulae are used to express policies. A conjunctive semantic formula $F^o_{\boldsymbol{v}} = \bigwedge_{i=0}^{n} \phi_i$ over an ontology o is a conjunction of atomic assertions ϕ_i, where a vector of variables used in these assertions is represented by $\boldsymbol{v} = \langle ?x_0, \ldots, ?x_n \rangle$. For the sake of convenience $\bigwedge_{i=0}^{n} \phi_i \equiv \{\phi_1, \ldots \phi_n\}$

refers to a conjunctive formula. Based on this, F^o_v can be considered as $T^o_v \cup R^o_v \cup C^o_v$, where T^o_v is a set of type assertions using the concepts from o, e.g. $\{student(?x_i), nurse(?x_j)\}$; R^o_v is set of relation assertions using the relations from o, e.g. $\{marriedTo(?x_i, ?x_j)\}$ and C^o_v is a set of constraint assertions on variables. Each constraint assertion is of the form $?x_i \lhd \beta$, where β is a constant and $\lhd$ is one of the symbols $\{>, <, =, \neq, \geq, \leq\}$. A constant is either a data literal (e.g. a numeric value) or an individual defined in o.

Variables are divided into two categories: data-types and object variables. A data-type variable refers to data values (e.g. integers) and can be used only once in R^o_v. On the other hand, an object variable refers to individuals (e.g. University_of_Aberdeen) and can be used any number of times in R^o_v. Equivalence and distinction between the values of object variables can be defined using OWL properties *sameAs* and *differentFrom* respectively, e.g. *owl:sameAs(?x,?y)* denotes that variables *?x* and *?y* refer to the same individual. In the rest of the paper, the symbols α, ρ, φ, and e are used as a short hand for semantic formulae.

Definition 1. Given an ontology o, a conditional policy π is defined as $\alpha \longrightarrow N_{\chi:\rho}\left(\lambda : \varphi\right)/e$

- α, a conjunctive semantic formula, is the activation condition of the policy.
- $N \in \{O, P, F\}$ indicates the modality of the policy (i.e., if the policy is an obligation, permission or prohibition).
- χ is the policy addressee and is described by ρ using only the *role* concepts from the ontology (e.g. $?x : Doctor(?x) \wedge Female(?x)$, where *Doctor* and *Female* are defined as sub-concepts of the concept *role* in the ontology). That is, ρ is of the form $\bigwedge_{i=0}^{n} r_i(\chi)$, where $r_i \sqsubseteq role$. Note that χ may directly refer to a specific individual (e.g. *John*) in the ontology or a variable.
- $\lambda : \varphi$ is the regulated action or state. λ is a variable referring to an action or a state that is regulated by the policy, where λ, is described by φ using the concepts and properties from the ontology. For example, $?a : UpdateHealthRecord(?a) \wedge hasPatient(?a, John) \wedge hasRecord(?a, HealthReport218$ $.pdf)$, where *UpdateHealthRecord* is an *action* concept. Each action concept has only a number of functional relations (aka. functional properties) [1] and these relations are used while describing an instance of that action.
- e is the expiration condition of the policy.

∎

An example of an RDF representation of an OWL-POLAR policy ***'Doctors are permitted to access patients' records.'*** is given below.

```
<policy>
<var> ?x </var>
<var> ?a </var>
<var> ?b </var>
<addressee> ?x </addressee>
```

```
<role> Doctor(?x) </role>
<modality> P </modality>
<action var = "?a"> Access(?a), PatientRecord(?b),
about(?a,?b), hasActor(?a,?x)</action>
</policy>
```

As stated in Section 1 conflicts are practically unavoidable. Effective conflict detection and resolution mechanisms are required for authoring consistent, unambiguous and functional policies. Here, following Castelfranchi [5], we distinguish between logical and functional conflicts.

Definition 2. Logical Conflicts arise between policies when the same action is both prohibited and permitted or both prohibited and obligated at the same time. Entities adopting logically conflicting policies will not necessarily be able to decide which policy should be respected. To avoid such conflicts, a thorough understanding of the exact meaning and the implications of a policy, individually and as a part of a set of policies, is required.

Given logically conflicting policies $\pi_i = \alpha^i \longrightarrow A_{\chi^i:\rho^i}\left(\lambda^i : \varphi^i\right)/e^i$ and $\pi_j = \alpha^j \longrightarrow B_{\chi^j:\rho^j}\left(\lambda^j : \varphi^j\right)/e^j$, The logical conflict L is denoted as : $L \longmapsto \pi_i \times \pi_j$. ▮

For example, consider two policies defined in healthcare domain.

– $\pi 1$ - Doctors are prohibited from modifying the chemotherapy regime of a patient.
– $\pi 2$ - Oncology specialists are permitted to modify the chemotherapy regime of a patient.

Since an Oncology Specialist is a subclass of Doctor, both the policies $\pi 1$ and $\pi 2$ are applicable. Hence, a logical conflict exists between $\pi 1$ and $\pi 2$. This conflict arises due to $\pi 1$ being ill-formed; it should refer to Doctors who are not oncology specialists.

Definition 3. Functional Conflicts arise between policies and underlying goals of an organisation [5]. For example, a policy requires some action to be performed that has a side-effect (or a pre-condition) that is harmful to a goal of the organisation.

Given functionally conflicting policies $\pi_i = \alpha^i \longrightarrow A_{\chi^i:\rho^i}\left(\lambda^i : \varphi^i\right)/e^i$ and $\pi_j = \alpha^j \longrightarrow B_{\chi^j:\rho^j}\left(\lambda^j : \varphi^j\right)/e^j$, The functional conflict F is denoted as : $F \longmapsto (\pi_i \times \pi_j)_{type}$, where *type* denotes the type of the conflict such as forbidden side-effect, forbidden pre-condition, inaccessible input/output, etc. ▮

Consider, for example, two policies defined in healthcare domain.

– $\pi 1$ - Nurses are permitted to perform various tests on patients
– $\pi 2$ - Nurses are prohibited from updating health-details of patients.

As a side-effect of performing various tests, nurses update health-records of patients. Hence, a functional conflict (due to a side-effect) exists between $\pi 1$ and $\pi 2$.

Let us consider another example.

– $\pi 1$ - Doctors are obliged to study cases of patients.
– $\pi 2$ - Doctors are prohibited from accessing complete medical history of patients.

Access to complete medical history of a patient is required as a pre-condition for a thorough study of a case. Hence, a functional conflict (due to pre-condition) exists between $\pi 1$ and $\pi 2$.

Conflict detection and resolution are computationally expensive. Hence, the reasoning mechanism must focus on the most relevant conflicts, given the goals of the organisation/agent. Conflicts occurring in a system are of varying significance. Some conflicts have higher chances of occurring and they need to be resolved statically, while others have very rare chances of occurring and they can be resolved at runtime [7]. We argue that organisational goals provide the appropriate focus for identifying more critical conflicts.

Given a specific domain, and given a set of organisational goals, the circumstances in which a certain conflict arises may be highly unlikely, albeit possible. What are the benefits of resolving such a conflict? Potentially, marginal. Moreover, policies are designed by people, for people, and hence they need to be understandable with respect to given context. Thus, it is important to asses if a set of policies that covers all possibilities in a consistent way is necessary, or whether it is better to live with some inconsistency for the sake of accessibility of organisational policies to human actors. For example, an agent in role x can perform α and agent in role y cannot. There is nothing in the ontology that states that memberships of these roles are disjoint (i.e., an individual could play both). This conflict may only be relevant, for example, when the goals of the organisation require α, and a number of individuals who can perform α play both roles x and y. It may be the case that this conflict need not be resolved otherwise. Hence, it is crucial to identify conflicts, the resolution of which, will lead to maximum benefits, satisfy relevant safety constraints and promote goal achievement.

4 Activity Prioritisation Model

The aim of the activity prioritisation model is to identify activities that are most important given the domain and goals of an organisation. Our model is based on the Page Rank algorithm [3], an algorithm that assigns a numeric weight to each element of a hyperlinked set of documents with the purpose of measuring its relative importance within the set. Page Rank score is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. The higher the number of links to a page the higher its Page Rank score. A hyperlink to a page is counted towards its score. Outbound link from a page gets a portion of the Page Rank score that is equal to the Page Rank score of the page divided by the total number of outbound links.

We have a set of activities connected by transitions. The transitions are defined by possible plans to achieve a goal. Our requirement is to rank the activities and define how likely it is that an activity will be performed while achieving the goal. Similar to multiple incoming links to a page in the Page Rank algorithm, we can have an activity that is a part of multiple plans. Further, activities that have higher probability of successful execution and that help in achieving goals with lower cost are more likely to be chosen for execution. Hence, unlike Page Rank algorithm, the average cost of achieving the goal after performing the activity and probability of success associated with it must also be considered while defining the rank of the activity. In our adaptation of the Page Rank algorithm, the priority (i.e., rank) of each activity is computed using the probability of the activity being executed while achieving a goal, the average cost of achieving the goal from the activity, and the probability of successful execution of the activity.

Consider an organisational goal G, and an initial situation T. We assume that a goal and initial situation are conjunctions of states, each representing a partition of all possible states. We denote A_c to be the set of all atomic actions defined in the domain and $Plans$ to be the set of all plans that will, according to the action specifications transform the world from the initial situation to achieve the goal.

Definition 4. A plan $p \in Plans$ is a sequence of atomic actions $(\alpha_1, \alpha_2, ..., \alpha_n)$ such that $\{\alpha_1, \alpha_2, ..., \alpha_n\} \in A_c$. ▮

Each atomic action (or activity) is annotated with information about the activity thus:

Definition 5. An annotation τ_i for action α_i is a tuple $< A_i^{parents}$, $A_i^{children}$, $\Upsilon_{i \to g}$, w_i^o, w_i^u, $\pi_i >$, where $A_i^{parents}$ is the set of all actions preceding α_i in all the plans containing α_i; $A_i^{children}$ is the set of all actions that follow α_i in all the plans containing α_i and $\Upsilon_{i \to g}$ is a set of costs of every path from α_i to α_g, where α_g is an activity in the plan that achieves goal G. The original and ultimate weight of α_i are represented by w_i^o and w_i^u respectively (see below for how these are computed). π_i is the active policy (i.e., activation condition is true and expiration condition is false) that regulates α_i. ▮

Policy annotations are performed by comparing action descriptions and inputs of every active policy with the activity. If they match then the policy is annotated to the activity. If no such active policy exists (i.e., if no active policy regulates the activity) a temporary active permission is generated that expires when the goal is achieved.

The priority of an activity within a set is computed in two stages. First, the original weight of an activity is computed, such that it is inversely proportional to the average cost of achieving the goal (i.e., the lower the average cost of achieving the goal the higher the weight of the activity), and directly proportional to the probability of successful execution of the activity (i.e., the higher the probability higher the weight of the activity). The assumption here is that low cost/high probability plans have higher precedence over high cost/low probability plans. The formulae for calculating the original weight of an activity α_i are:

$$\Upsilon_{i\to g}^{avg} = \frac{\sum \Upsilon_{i\to g}}{|\Upsilon_{i\to g}|} \tag{1}$$

$\Upsilon_{i\to g}^{avg}$ is the average cost of achieving goal G from α_i (i.e., the average of the values stored in $\Upsilon_{i\to g}$).

$$\Upsilon^{avg} = \sum_{\alpha_i \in \bigcup\limits_{p \in Plans} p} \Upsilon_{i\to g}^{avg} \tag{2}$$

Υ^{avg} is the sum of average costs of achieving the goal for all the activities being considered

$$w_{\alpha_i}^{o} = Pr(\hat{\alpha_i}) \times \left(\frac{\Upsilon^{avg}}{\Upsilon_{i\to g}^{avg}} \right) \tag{3}$$

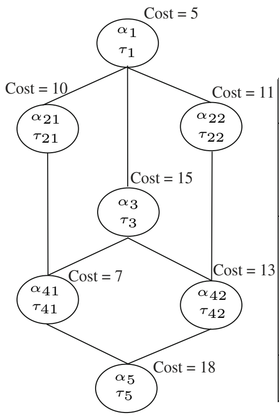
$Pr(\hat{\alpha_i})$ represents the probability of successful execution of α_i and the ratio represents relative importance of an activity within a set in terms of cost of achieving the goal.

The number of plans that the activity is part of and the weight the activity inherits from its preceding activities is not considered in the original weight. After determining original weights of all the activities, therefore, the ultimate weight of each activity is computed. This ultimate weight takes into account the weight induced by preceding (or parent) activities. The formulae for calculating the ultimate weight of an activity α_i are:

$$w_\alpha^u = w_\alpha^o \qquad \textit{if} \quad \left|A_\alpha^{parents}\right| = 0 \tag{4}$$

$$= \sum_{\beta \in A_\alpha^{parents}} \left(w_\beta^u \times \frac{w_\alpha^o}{\sum\limits_{\epsilon \in A_\beta^{children}} w_\epsilon^o} \right) \qquad \textit{otherwise} \tag{5}$$

In the Page Rank algorithm, the score given to a page is distributed to all its outbound links. In our model, the ultimate weight of an activity is distributed in proportion to the original weights (i.e., the relative importance within the set of activities) of its child activities. If α is the first activity in a plan, then the ultimate weight remains the same as the original weight of α. If α is not the first activity in the plan, the ultimate weight of activity α is the sum of the weights induced by all its preceding activities, $A_\alpha^{parents}$. A portion of the ultimate weight of these parent activities is induced on each activity that follows in proportion to the original weight of the child activity. If a parent has only one child activity the ultimate weight of the parent is induced on the child. If a parent has more than one children activities, then its weight is divided among them in proportion to their original weights. Activities are then prioritised according to their ultimate



	$\Upsilon_{i\to g}$	$\Upsilon^{avg}_{i\to g}$	$Pr(\hat{\alpha})$	ϖ^o_i	ϖ^u_i	Priority	ϖ^o_i	ϖ^u_i	Priority
α_1	40,45, 51,47	45.75	0.85	4.454372	4.454372	2	6.8383594	6.8383594	1
α_{21}	35	35	0.48	3.2879999	1.3482656	6	0.9296622	1.144525	7
α_{22}	42	42	0.78	4.4525	1.8257765	5	2.3911257	2.9437609	4
α_3	40,46	43	0.56	3.1223254	1.2803297	7	2.2337992	2.7500732	5
α_{41}	25	25	0.77	7.3842998	2.0073442	4	4.5237107	2.2811263	6
α_{42}	31	31	0.9	6.9604836	2.4470277	3	6.4216723	4.557233	3
α_5	18	18	0.9	11.987499	4.454372	1	18.403233	6.838359	2

$\sum \Upsilon^{avg}_{i\to g} = 239.75$

Fig. 2. Activity Annotation and Activity Prioritisation Model

weights. The higher the ultimate weight, the higher the priority. Original weight is used to rank two activities with the same ultimate weights.

Fig 2 shows an example of activity prioritisation. The first activity in the plan, α_1 has an ultimate weight equal to its original weight. The ultimate weight of α_1 is divided among its child activities α_{21}, α_{22} and α_3 in proportion to their original weights. Since activities α_{21}, α_{22}, α_{41} and α_{42} have only one child activity each, their weights are not divided. The ultimate weight of α_{41} is the sum of its share from the ultimate weight of α_3 and the ultimate weight of α_{21}. Similarly, the ultimate weight of α_{42} is computed from a share of the ultimate weight of α_3 and the ultimate weight of α_{22}. Finally, the ultimate weight of α_5 is the sum of the ultimate weights of α_{41} and α_{42}. The ultimate weights of α_1 and α_5 are approximately the same. Hence, their priorities are decided according to their original weights.

If we were interested in simply prioritising activities in isolation, this adaptation of the Page Rank algorithm would be sufficient. However, we are also interested in prioritising conflict resolution in order to find good plans to achieve organisational goals. In other words, along with identifying high-rank activities, our aim is to highlight an optimum (i.e., higher probability and lower cost) plan towards the goal. For this reason, we now present an adaptation to the method for computing the original weight of an activity that takes these factors into account. In addition to the probability of successful execution of an activity, we need to consider the probabilities of successful execution of all the plans of which the activity is a part.

$$Pr(\hat{p}) = \prod_{\alpha \in p} Pr(\hat{\alpha}) \tag{6}$$

Probability of success of a plan is a product of probabilities of success of all the activities in that plan. Now we can reformulate our method for computing the original weight of an activity by biasing it toward activities that occur in plans that are more likely to succeed.

$$w^o_{\alpha_i} = Pr(\hat{\alpha}_i) \times \sum_{p \in PlansWith(\alpha_i)} Pr(\hat{p}) \times \left(\frac{\Upsilon^{avg}}{\Upsilon^{avg}_{i \to g}} \right) \tag{7}$$

Where $PlansWith(\alpha_i) = \{p : p \in Plans \wedge \alpha_i \in p\}$

The formula for computing $w^u_{\alpha_i}$ (i.e., the ultimate of activity α_i) remains the same.

The last three columns in Fig. 2 show the outcome using the refined activity prioritisation model, where a plan with highest probability $\{\alpha_1, \alpha_{22}, \alpha_{42}, \alpha_5\}$ is highlighted first. We evaluated the heuristic activity prioritisation model by generating random plans (i.e., random directed acyclic graphs with varying number of nodes within the range of 5 to 25). Each activity was assigned a random cost and probability of success. Preliminary results are shown in Fig 3 and 4. Fig 3 shows the frequency of the best, the second best, etc. plan being highlighted first. Fig 4 shows the average ratio of the number of activities analysed in order to identify the first feasible plan to the average plan length. These preliminary results are promising, where the prioritisation heuristic tends to highlight plans with lower cost and higher probability of success. Furthermore, the average number of activities considered tends to be within 1 and 2 times the average number of activities in a plan (i.e, average plan length). However, this ratio increases as the number of activities increases and so further investigation is required.

Activity prioritisation process requires four iterations on the given set of activities. In first iteration policy annotations on activities are performed and sets for preceding and following activities are computed. Initial weights and ultimate weights are calculated in second and third iterations. Finally, priorities are assigned in fourth iteration. Hence, the computational complexity of the activity prioritisation process is linear with respect to total number of activities, i.e., $\mathcal{O}(4n)$, where n is the total number of activities. Moreover, only conflicts that involve the policies annotated to high-rank activities will be detected and resolved. All the policies in the set of policies are not checked against each other. Also, if the plans are static activity prioritisation will be performed only once.

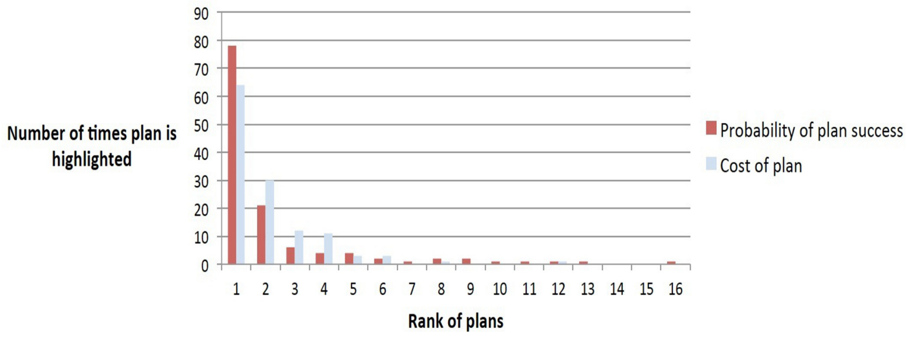


Fig. 3. Frequency of highlighting the best viable plan

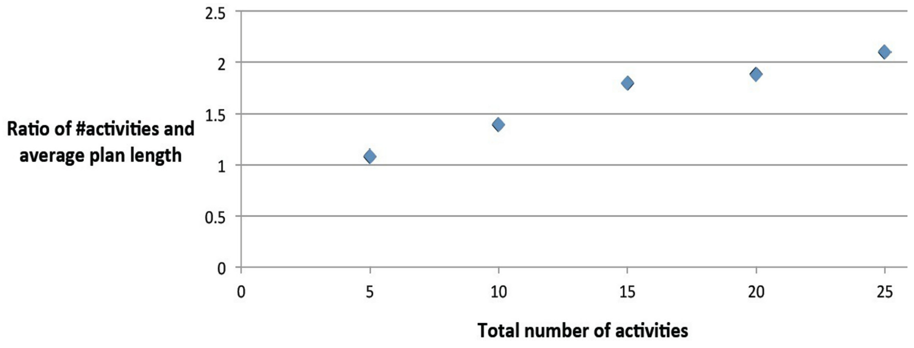


Fig. 4. Activities analysed to highlight a viable plan

5 Conflict Detection

We now have a heuristic mechanism for prioritising conflict detection and resolution. As described in the Definition 5 each activity is annotated with a policy that permits its execution. We may now reason about potential logical and functional conflicts. Following our ranking of activities, we ask the question: is a permission to perform the action consistent with our organisational policies?

We have extended the OWL-POLAR reasoning mechanism [6] by implementing algorithms to detect functional conflicts arising due to side-effects (i.e., an action has a side-effect that is prohibited by some other policy) and pre-conditions of actions (i.e., a pre-condition of an action cannot be fulfilled due to some prohibition).

Consider two policies $\pi_i = \alpha^i \longrightarrow A_{\chi^i:\rho^i} \left(\lambda^i : \varphi^i\right) / e^i$ and $\pi_j = \alpha^j \longrightarrow B_{\chi^j:\rho^j} \left(\lambda^j : \varphi^j\right) / e^j$. These policies will conflict with each other if the necessary conditions are satisfied; i.e., modalities of π_i and π_j are conflicting (Condition 1), and π_i and π_j are active for the same policy addressee in the same state of the world Δ (Conditions 2, 3 and 4).

1. A conflicts with B. That is, $A \in \{O, P\}$ while $B \in \{F\}$ (i.e, $(\lambda^i : \varphi^i)$ is permitted or obligated action and $(\lambda^j : \varphi^j)$ is a forbidden state).
2. There exists a substitution σ_i s.t. $\Delta \vdash \left(\alpha^i \wedge \rho^i\right) \cdot \sigma_i$, but no substitution σ_i' s.t. $\Delta \vdash \left(e^i \cdot \sigma_i\right) \cdot \sigma_i'$. In the state of the world Δ there exists at least one individual (substitution σ_i) in role ρ^i such that the activation condition α^i is true but the expiration condition e^i is not true.
3. There exists a substitution σ_j s.t. $\Delta \vdash \left(\alpha^j \wedge \rho^j\right) \cdot \sigma_j$, but no substitution σ_j' s.t. $\Delta \vdash \left(e^j \cdot \sigma_j\right) \cdot \sigma_j'$. In the state of the world Δ there exists at least one individual (substitution σ_j) in role ρ^j such that the activation condition α^j is true but the expiration condition e^j is not true.
4. $\chi^i \cdot \sigma_i = \chi^j \cdot \sigma_j$. The substitutions σ_i and σ_j are such that the individuals χ^i and χ^j are the same.

Algorithm 1 checks these necessary conditions. Two policies π_i and π_j as specified above and the type of functional conflict that is being checked for, are inputs to the algorithm. The first step is to test if A conflicts with B (line 2). If they are conflicting, a canonical state of the world Δ, in which π_i is active, is created by freezing[1] $\left(\alpha^i \wedge \rho^i\right)$ with a substitution σ_i, mapping the variables in $\left(\alpha^i \wedge \rho^i\right)$ to the fresh individuals in Δ (line 3). Δ is then queried with $\left(\alpha^i \wedge \rho^i\right)$ (line 4). The results of this query satisfy $\left(\alpha^i \wedge \rho^i\right) \cdot \sigma_i$. For each σ_k satisfying $\left(\alpha^i \wedge \rho^i\right) \cdot \sigma_i$, Δ is updated by freezing $\left(\alpha^j \wedge \rho^j\right) \cdot \sigma_k$, without removing any individual from its existing *ABox* (line 6). As a result of this process, σ_j is the substitution mapping the variables in $\left(\alpha^j \wedge \rho^j\right) \cdot \sigma_k$ to the new fresh individuals in the updated Δ, so that $\chi^i \cdot \sigma_i = \left(\chi^j \cdot \sigma_k\right) \cdot \sigma_j$. The consistency of the resulting state of the world Δ is tested (line 7). If this is not consistent, it is concluded that it is not possible to have a state of the world satisfying the requirements. If the resulting

Algorithm 1. Anticipate if π_i may functionally conflict with π_j.

```
 1: Input:   Policy $\pi_i = \alpha^i \longrightarrow A_{\chi^i:\rho^i}\left(\lambda^i : \varphi^i\right)/e^i$,
             Policy $\pi_j = \alpha^j \longrightarrow B_{\chi^j:\rho^j}\left(\lambda^j : \varphi^j\right)/e^j$
             $type$
 2: if ( $(A \in \{O, P\}$ and $B \in \{F\})$ then
 3:    $\langle\Delta, \sigma_i\rangle = freeze(\alpha^i \wedge \rho^i)$
 4:    $rs \quad = query(\Delta, \alpha^i \wedge \rho^i)$
 5:    for all $(\sigma_k \in rs)$ do
 6:       $\langle\Delta, \sigma_j\rangle = update(\Delta, \left(\alpha^j \wedge \rho^j\right) \cdot \sigma_k)$
 7:       if $(isConsistent(\Delta))$ then
 8:          if $(query(\Delta, e^i \cdot \sigma_i) = \emptyset$ and $query(\Delta, \left(e^j \cdot \sigma_k\right) \cdot \sigma_j) = \emptyset)$ then
 9:             if $(type = sideeffect)$ then
10:                return $checkSideEffects(\lambda^i, \lambda^j, \sigma_k, \Delta)$
11:             end if
12:             if $(type = precondition)$ then
13:                return $checkPreconditions(\lambda^i \cdot preconditions, \lambda^j, \sigma_k, \Delta)$
14:             end if
15:          end if
16:       end if
17:    end for
18: end if
19: return false
```

[1] In order to test whether qA subsumes qB, the standard technique of query freezing is used to reduce query containment problem to query answering in Description Logics [6]. In this technique a canonical knowledge-base is built from the query by replacing variables in the query with fresh individuals, adding each individual appearing in the query to the canonical knowledge-base and inserting relationships between individuals and constants defined in the query into the canonical knowledge-base. As a result of this process, the canonical knowledge-base contains a pattern that exists only in ontologies that satisfy the query.

Δ is consistent, the expiration conditions of the policies are checked. If both policies are active in the resulting state of the world (line 8), we test for the sufficient conditions. If policies are being checked for functional conflicts arising due to side-effects then (line 9), the function 'checkSideEffects' is called and the value returned by the function is returned (line 10). If policies are being checked for functional conflicts arising due to pre-conditions then (line 12), the function 'checkPreconditions' is called and the value returned by the function is returned (line 13). If any of these requirements do not hold then *false* is returned (line 19). The functions 'checkSideEffects' and 'checkPreconditions' are explained in the subsequent algorithms.

Algorithm for Detecting Functional Conflicts Arising Due to Side-Effects: The policies π_i and π_j may conflict functionally (due to side-effects) if, in addition to the necessary conditions stated above, the following sufficient condition is also satisfied.

– λ^j is the effect of performing λ^i (Reasoning about an atomic action λ^i and a state λ^j).

Algorithm 2. Check Side Effects

```
1: Input: $\lambda^i, \lambda^j, \sigma, \Delta$
2: $initial = clone(\Delta)$
3: $state1 = query(initial, \lambda^j \cdot \sigma)$
4: $next = applyActionToState(\lambda^i \cdot \sigma, initial)$
5: $state2 = query(next, \lambda^j \cdot \sigma)$
6: if $((state1 = \emptyset)$ and $(state2 \neq \emptyset))$ then
7:     return true
8: end if
9: return false
```

Algorithm 2 is used to check side-effects of an atomic task. Permitted/obligated action (λ^i), forbidden state (λ^j), substitution (σ) for which both the policies are active for the same individuals and the state of the world (Δ) where both the policies are active for the same individuals at the same time, are inputs. The state of the world (Δ) is cloned and the resulting state of the world is stored in 'initial' (line 2) . This clone is queried for ($\lambda^j \cdot \sigma$) and the results are stored in the set 'state1' (line 3). The atomic action ($\lambda^i \cdot \sigma$) is then applied to 'initial' (line 4) and the resulting state of the world is stored in 'next'. The state of the world 'next' is then queried for ($\lambda^j \cdot \sigma$) and the results are stored in the set 'state2' (line 5). If 'state1' is empty and 'state2' is not empty then it is concluded that λ^j is an effect of performing λ^i and *true* is returned, otherwise the function returns *false*.

Algorithm for Detecting Functional Conflicts Arising Due to Preconditions: The policies π_i and π_j may conflict functionally (due to preconditions) if, in addition to the necessary conditions stated above, the following sufficient condition is also satisfied.

– λ^j is a precondition of λ^i (Reasoning about an atomic action λ^i and a state λ^j).

Algorithm 3 is used to check pre-conditions. A conjunction query of preconditions of the permitted / obligated action (*preconditions*), forbidden state (λ^j), substitution (σ) for which both the policies are active for the same individuals and the state of the world (Δ) where both the policies are active for the same individuals at the same time, are inputs. The state of the world (Δ) is cloned (line 2) the resulting state of the world is stored in 'initial'. This clone is updated by freezing ($preconditions \cdot \sigma$) and the resulting state of the world is stored in 'previous' (line 3). The state of the world 'previous' is queried for ($\lambda^j \cdot \sigma$) (line 4) and results are stored in the set 'state'. If 'state' is not empty then, it is concluded that one of the preconditions of the permitted/obligated action (i.e., one of the elements of *preconditions*) is the same as the forbidden state λ^j and *true* is returned (line 6), otherwise the function returns *false* (line 8).

Algorithm 3. Check Preconditions

```
1: Input: $preconditions$, $\lambda^j$, $\sigma$, $\Delta$
2: $initial = clone(\Delta)$
3: $\langle previous, \_\rangle = update(initial, preconditions \cdot \sigma)$
4: $state \quad = query(previous, \lambda^j \cdot \sigma)$
5: if ($state \neq \emptyset$) then
6:     return true
7: else
8:     return false
9: end if
```

6 Discussion

In this paper, we have presented a heuristic method for prioritising reasoning about policy conflicts in such a way that the most preferred plans to achieve organisational goals are explored as early as possible, and presented both some preliminary evaluations of this heuristics and algorithms for detecting functional conflicts. This is, of course, only a part of the story, and effective conflict resolution is an essential part of a complete solution. Various strategies can be used to resolve conflicts, e.g. adding a new policy, modifying action constraints of a policy, modifying the activation window of a policy, norm-curtailment [15], and prioritising policies. However, addition of new policies or modification of the conflicting policies might introduce new conflicts. Hence, a new/modified policy must be checked for consistency against all existing policies. Prioritising technique does not make any changes to the existsing set of policies. Hence, it is guaranteed not to introduce new conflicts. Checking the whole set of policies that are already reasoned about is not required.

Classic forms of policy prioritisation are lex-superior (policies/norms from higher authority take precedence) and lex-posterior (most recent policies/norms take precedence) [9]. In lex-specialis, a generic policy is overridden by a more specific one. All these techniques, however, do not permit a resolution that involves different policies taking precedence in different contexts. Also, it is not necessarily appropriate to apply one resolution method for all conflicts in a system [7]. User-defined precedence may play an important role in resolving conflicts.

Extending OWL-POLAR further, we have implemented a policy prioritisation method for resolving conflicts and refining policies. In this method policies can be prioritised based on cost of violation or specificity or user-input. We have extended Definition 1 to include Π - a list of references (e.g. policy names) to overriding policies. Hence, references to overriding policies are maintained by overridden policy. Consider two conflicting policies $\pi_i = \alpha^i \longrightarrow A_{\chi^i:\rho^i}\left(\lambda^i : \varphi^i\right)/e^i/\Pi^i$ and $\pi_j = \alpha^j \longrightarrow B_{\chi^j:\rho^j}\left(\lambda^j : \varphi^j\right)/e^j/\Pi^j$. Where Π^i and Π^j are the sets containing references to overriding policies. Let us consider that after using one of the techniques of policy prioritisation described above, policy π_i overrides π_j. Hence, Π^j will be modified as $\Pi^j = \{\pi_i\}$. Consider that policy $\pi_k = \alpha^k \longrightarrow C_{\chi^k:\rho^k}\left(\lambda^k : \varphi^k\right)/e^k/\Pi^k$ also conflicts with π_j and π_k overrides π_j. Hence, Π^j will be modified as $\Pi^j = \{\pi_i, \pi_k\}$. Conflict detection mechanism of OWL-POLAR is augmented so that resolutions are taken into account.

In future, we will extend the OWL-POLAR further to detect other types of functional conflicts, e.g. inaccessible input/output and incoherent domain. We also plan to evaluate the activity prioritisation model with human users. We will investigate if activity prioritisation model identifies an optimum set of conflicts to resolve, such that the goal is achieved with likelihood $\geq$ threshold. It will be investigated and evaluated if the automated support mechanisms developed for helping policy authors in resolving policy conflicts and refining policies can also aid collaborative authoring and refinement. Currently, we are annotating a single active policy to an activity and considering only static plans. In future, we will investigate techniques for annotating multiple policies to an activity ensuring consistency of annotated policies. We will also improve our model to incorporate dynamically changing plans.

7 Conclusion

The key questions addressed in this paper are: how to identify conflicts that are most relevant given the domain and goals of organisation, and how to resolve conflicts that are most important. The activity prioritisation model is used to identify the most significant activities that can be performed while achieving a goal. Active policies that regulate high priority activities are reasoned about to detect logical and functional conflicts. This ensures identification of conflicts that are most relevant to the domain and goals of organisation. Only those conflicts that have very high chances of impairing the goal-achievement (i.e., conflicts involving high priority activities) will be resolved using the conflict resolution techniques described in the paper. Preliminary results indicate that along with identifying the most crucial activities, the refined activity prioritisation model emphasises on highlighting an optimum plan (i.e., higher probability and lower cost) for achieving a goal. Depending on significance of goal, resource availability and computational capacity a threshold can be defined for number of activities to be considered for conflict detection and resolution. A threshold for number of conflicts that can be resolved offline can also be defined. Thus, fewer policy violations and decreased possibility of failures in plan execution at runtime are ensured by the activity prioritisation model.

References

1. W. O. W. Group, OWL 2 Web Ontology Language: Document overview, `http://www.w3.org/TR/owl2-overview`
2. Bordini, R.H., Hübner, J.F., Vieira, R.: Jason and the golden fleece of agent-oriented programming. In: Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E. (eds.) Multi-Agent Programming: Languages, Platforms and Applications, pp. 3–37. Springer (2005b)
3. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems 30(1-7), 107–117 (1998)
4. Broersen, J., Dastani, M., Hulstijn, J., Huang, J., van der Torre, L.: The BOID Architecture - Conflicts Between Beliefs, Obligations, Intentions and Desires. In: Proceedings of the 5th International Conference on Autonomous Agents, pp. 9–16. ACM Press (2001)
5. Castelfranchi, C.: Formalizing the Informal?: Dynamic Social Order, Bottom-Up Social Control, and Spontaneous Normative Relations. Journal of Applied Logic 1(1-2), 47–92 (2003)
6. Şensoy, M., Norman, T.J., Vasconcelos, W.W., Sycara, K.: OWL-POLAR: Semantic Policies for Agent Reasoning. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 679–695. Springer, Heidelberg (2010)
7. Dunlop, N., Indulska, J., Raymond, K.: Methods for Conflict Resolution in Policy-Based Management Systems. In: Proceedings of the 7th International Enterprise Distributed Object Computing Conference, pp. 98–109. IEEE Computer Society (2003)
8. Johnson, M., Karat, J., Karat, C.-M., Grueneberg, K.: Usable Policy Template Authoring for Iterative Policy Refinement. In: Proceedings of the 2010 IEEE International Symposium on Policies for Distributed Systems and Networks, pp. 18–21. IEEE Computer Society (2010)
9. Kollingbaum, M.J.: Norm-Governed Practical Reasoning Agents. Ph.D. thesis, Dept. of Computing Science, University of Aberdeen (2005)
10. Meneguzzi F.: Extending Agent Languages for Multiagent Domains. Ph.D. thesis, University of London, King's College London (2009)
11. Oh, J., Meneguzzi, F., Sycara, K., Norman, T.J.: An Agent Architecture for Prognostic Reasoning Assistance. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI, pp. 2513–2518 (2011)

12. Sycara, K., Norman, T.J., Giampapa, J.A., Kollingbaum, M.J., Burnett, C., Masato, D., McCallum, M., Strub, M.H.: Agent Support for Policy-Driven Collaborative Mission Planning. The Computer Journal 53(5), 528–540 (2009)
13. Tonti, G., Bradshaw, J.M., Jeffers, R., Montanari, R., Suri, N., Uszok, A.: Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 419–437. Springer, Heidelberg (2003)
14. Uszok, A., Bradshaw, J.M., Breedy, M.R., Bunch, L., Feltovich, P., Johnson, M., Jung, H.: New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS. In: 2008 IEEE Workshop on Policies for Distributed Systems and Networks, pp. 145–152. IEEE Computer Society (2008)
15. Vasconcelos, W.W., Kollingbaum, M.J., Norman, T.J.: Normative Conflict Resolution in Multi-Agent Systems. Auton Agent Multi-Agent Systems 19(2), 124–152 (2009)

Norms as Objectives: Revisiting Compliance Management in Multi-agent Systems

Aditya Ghose[1] and Tony Bastin Roy Savarimuthu[2]

[1] Decision Systems Laboratory
School of Computer Science and Software Engineering
University of Wollongong, NSW 2522 Australia
aditya@uow.edu.au
[2] Dept. of Information Science
University of Otago

Abstract. This paper explores a hitherto largely ignored dimension to norms in multi-agent systems: the normative role played by optimization objectives. We introduce the notion of *optimization norms* which constrain agent behaviour in a manner that is significantly distinct from norms in the traditional sense. We argue that optimization norms underpin most other norms, and offer a richer representation of these. We outline a methodology for identifying the optimization norms that underpin other norms. We then define a notion of compliance for optimization norms, as well as a notion of consistency and inconsistency resolution. We offer an algebraic formalization of *valued optimization norms* which allows us to explicitly reason about degrees of compliance and graded sanctions. We then outline an approach to decomposing and distributing sanctions amongst multiple agents in settings where there is joint responsibility.

1 Introduction

The connection between norms and preferences and between norms and optimization objectives has received relatively little attention in the literature. Boer et al [1] have argued that legal norms may be viewed as statements of ceteris paribus preference. van der Torre and Tan [2] have defined a preference-based semantics for norms which have been leveraged by Dignum et al [3] to develop a semantic account of how goals and intentions are obtained from desires via a process constrained by norms and obligations (utilities are also mentioned in [4] but not related to norms).

The connections, however, are deeper and merit closer scrutiny. Let us consider the connections with preferences and optimization objectives first. Optimization problems are traditionally formulated via a set of *decision variables* (a complete assignment of values to these constitutes a *solution* to the problem), a set of *constraints* on these variables and an *objective function* formulated from these variables whose value we seek to *optimize* (i.e., maximize or minimize). Solutions which satisfy all of the applicable constraints are called *feasible* solutions, while those which optimize the value of the objective function are called *optimal* solutions. An objective function may be viewed as generating a preference relation on the set of feasible solutions, such that optimal solutions are the most preferred of the feasible solutions under this relation. Given a set of alternatives (in the this instance, the set of feasible solutions) an objective function may be viewed as an intensional representation of the underlying preference relation.

Imagine a future where organizations may be referred to "carbon tribunals" for non-compliance with the carbon-mitigation norm. We would argue that in its purest sense, this norm must be represented as the optimization objective *minimize carbon-footprint*. One might argue that in real-life, such norms are manifested as simpler numeric carbon mitigation targets for organizations (e.g. "reduce your cumulative emissions by n tons"). However, this representation of the norm is a compromise, where some regulatory authority has sought to present a simpler (to evaluate and understand) target for organizations to meet, by trading off the need to impose a target that achieves real emissions reduction against the need to not impose a norm that would be infeasible for organizations to meet. We would argue that such norms (with numerically specified targets) must be accorded a status befitting their true nature: as simplifications and imperfect, incomplete compromises between the true intent of the norm and the current business reality. The numeric target is a fragile construct, contingent on the perceived business context at the time the norm was formulated. The introduction of new technology may actually make a higher carbon-mitigation target feasible, but such a change would be hard to reflect dynamically on the norm (which would typically be reviewed and revised infrequently, together with other norms, by a regulatory authority). Similarly, difficult market conditions might oblige us to revise the carbon-mitigation targets downwards. Clearly, this constant need for revising the norm could be avoided if the norm were represented in its natural form (as the optimization objective *minimize carbon-footprint*).

It may appear at first blush that a notion of compliance with such norms is difficult to define. We will present evidence to the contrary. Imagine the organization charged with non-compliance with the carbon-mitigation offering its defence to the tribunal by presenting a log of all its key organizational decisions over an audit period and establishing that each choice of the organization from the available options was in fact the optimal one with respect to the *minimize carbon-footprint* objective. Intuitively, this is a reasonable defence, underpinned by the argument that the organization "did the best that it could" under the circumstances. We shall formalize a notion of compliance with norms that are optimization objectives along these lines.

Based on these considerations, we argue that there is a need to extend our ontology of norms with a new class of *optimization norms*. These are norms represented in the form of optimization objectives or, as we shall see later, as the preference relations that underpin them. Optimization norms do not admit boolean evaluation. We shall distinguish them from the more traditional conception of a norm (which does admit boolean evaluation) by referring to the latter as *boolean norms*. Our approach bears some similarity in underlying intuitions to studies of supererogation in deontic logic (see for instance [5]) but uses entirely different machinery.

Note that the use of optimization norms does not imply that there must exist a unique optimal state (or set of states) in the absolute sense. As the motivating example above illustrates, and as we shall discuss in greater detail later in this paper, optimization norms merely inform choice amongst the available feasible alternatives. That set of available feasible alternatives might change from context to context. This approach thus does not preclude dynamic contexts or norm evolution.

Part of our premise here is that "teasing out" the objective underpinning a norm and bringing it to bear on the reasoning process is important, for the following reasons. First, encoding norms as objective functions offers a more accurate and richer representation of norms. Second, it provides an opportunity to explicitly bring a mature body of results from the field of optimization to bear on norm-driven reasoning problems, opening up the possibility of significantly faster reasoners. Third, it permits us to relate norm-compliance to the notion of *satisficing* [6] of optimization objectives. Fourth, it enables us to define a notion of degrees of norm compliance, and correspondingly, graded sanctions.

This latter point is of particular importance, and can be analyzed from two perspectives. The first involves the notion of *graded compliance.* Many commonly occurring compliance requirements are stated in an imprecise fashion. Consider, for instance, the requirement *quarterly activity statements must be filed within a reasonable time frame* [7]. It is difficult to determine in a categorical fashion whether this requirement has been satisfied, given the ambiguity associated with determining whether a certain time frame is "'reasonable"'. One way of dealing with the problem is to "'contextualize"' such requirements through a (largely human-mediated) exercise of transforming these into crisp requirements by adding elements to the specific context (such as a definition of "'reasonable"' in a particular application context) [8]. Alternatively, one can make (potentially subjective) assessments of degrees of compliance. In the spirit of satisficing optimization objectives, thresholds on these degrees of compliance can be used to determine, for instance, whether the operations of an organization are sufficiently compliant, even in the absence of boolean assessments of compliance.

The second involves the related notion of *graded sanctions.* The use of formal reasoning tools to model, analyze and monitor contracts is becoming increasingly important [9] [10]. A particularly hard problem in this space is the formalization of sub-contracting and outsourcing (both increasingly common business practices). In particular, the decomposition of a set of penalties or sanctions amongst a set of sub-contractors is difficult to formalize. Intuitively, when a norm or contractual obligation is violated, we would expect the applicable penalties to be distributed amongst the sub-contractors in direct proportion to the extent of their contribution to (or responsibility for) the violation. While Villatoro et al [11] and Boella et al [12] have considered the problem of sanctions in multi-agent contexts, they have not described any machinery for decomposing a sanction to obtain individual agent-specific graded sanctions in settings where agents have joint responsibility (and hence graded levels of norm violation).

The remainder of this paper is structured as follows. In Section 2, we discuss how optimization norms underpin most traditional conceptions of norms (i.e., *boolean norms*), and provide methodological guidelines for how these might be identified/extracted from boolean norms. In Section 3, we motivate and define an algebraic formalization of optimization norms in the form of *valued optimization norms*. In Section 4, we identify two alternative notions of compliance with optimization norms, based on the extent of the horizon over which compliance is assessed. In Section 5, we identify alternative notions of consistency (both between optimization norms and between optimization and boolean norms). We also identify alternative approaches to the resolution of these inconsistencies. In Section 6, we address the problem of sanction management, and in particular, how sanctions might be decomposed, or distributed amongst a collection of agents that had shared responsibility for a norm (the violation of which leads to the sanctions in question). Section 7 involves a discussion of some related implementations that offer pointers to how a machinery for optimization norm enforcement might be implemented. We present concluding remarks in Section 8.

2 Identifying Optimization Norms

In a very intuitive manner, it is possible to articulate an objective function underpinning every norm. Consider the social norm that one should not litter. Given that there are many plausible extenuating circumstances where littering may in fact be permissible (e.g., one drops one's bag of sandwiches on the park bench to go prevent a child from stepping into traffic), the underpinning optimization objective is to *minimize* the extent of littering. In a social context, the articulation of the prohibition of littering may be viewed as a necessary simplification of the more complex (and more nuanced) underpinning social objective. In a similar vein, the social norm that prohibits delays in the payment of outstanding invoices is underpinned by the optimization objective to minimize the delay between the receipt of the invoice and payment. The social norm that obliges us to consult widely prior to taking decisions in organizational settings may be viewed as being underpinned by the optimization objective to maximize the extent of consultation prior to taking decisions. More generally, we could posit that:

- Prohibitions are underpinned by *minimization* objectives.
- Obligations are underpinned by *maximization* objectives.

The duality of maximization and minimization objectives (every maximization objective can be represented by a corresponding minimization objective and vice versa) is reflected by the duality of prohibitions and obligations [13]. In addition, fairness norms [14] can be encoded as load-balancing objectives.

In many cases, the boolean proposition associated with the norm can be transformed into a continuous valued variable. Thus, the boolean proposition *littering* is transformed into the variable *extent-of-littering*, the proposition *consultation* to the variable *extent-of-consultation* and so on in the examples above. In general, the following simple procedure might be used to identify the optimization objective underpinning a norm:

– Identify the action that the norm seeks to constrain. In our examples above, these would be "littering", "consultation" and "carbon emission".
– Identify a measure that is applicable on the task that norm makes reference to. In our examples above, these would be *extent-of-littering*, *extent-of-consultation* or *extent-of-carbon-emission*. These become the variables that the eventual objective function would refer to. In the following, we shall refer to this as the *task measure*.
– Identify whether the norm seeks to maximize or minimize the measure defined in the previous step (load balancing objectives can also be represented as maximization or minimization objectives). This gives us the final objective function.

Note that articulating an objective function does not automatically give us a fully formulated optimization problem (that would require the typically hard task of modeling constraints). However, as we shall see below, the optimality of choices with respect to an optimization objective can be evaluated using a variety of means even if the problem has not been formulated as an optimization problem.

It is important to note that our conception of optimization norms is not specifically intended for modeling social norms. The machinery we develop is applicable to both individual objectives (and preferences) as well as social norms and the preferences that underpin them.

It is also important to note that, our observations in this section notwithstanding, practical applications will likely involve both boolean and optimization norms. It is therefore important to develop machinery that can handle both (our discussion of norm consistency later in the paper will address the question of consistency between boolean and optimization norms).

3 An Alternative Formalization

In this section, we provide an alternative formalization of optimization norms in terms of an algebraic framework for preference handling and show how it offers a sophisticated machinery for dealing with *graded compliance* and correspondingly *graded sanctions*.

It is useful to examine first the space of alternative means for formalizing optimization norms. The most obvious is to represent optimization norms in the form of *objective functions* in the sense understood in the literature on operations research, or as *utility functions* in the sense of the literature on decision theory. One might also formalize these using preference relations of various kinds as formalized in the literature on preference handling. To the extent that an optimization norm needs to encode preference over states of affairs (or solutions) each of these approaches turn out to be equally viable (recall the discussion on the interplay between objective functions and preference in Section 1). Ultimately, our choice was guided by two additional requirements. First, the representation scheme needed to support an explicit notion of *degree of compliance*. Objective functions (or utility functions) arguably meet this requirement - the value of a (maximization) objective function for a given solution can be viewed as an indicator of how good that solution might be. Representations based on preference relations do not naturally lend themselves to analysis of degree of compliance. Second, the representation scheme had to be general enought to admit assessments of preference on multiple heterogeneous scales (as is likely to be the case in real-life applications) including both quantitative and qualitative scales. Both objective/utility functions and preference relation based approaches fall short relative to this requirement.

The c-semiring framework [15] was chosen for our formalization of optimization norms primarily because it satisfies all of the requirements discussed above. The framework was originally developed for defining soft constraints as preferences over assignments of values to decision variables in (potentially over-constrained) constraint satisfaction problems. For our purposes, the c-semiring framework enables abstract encodings of preference over multiple heterogeneous scales (which could be both qualitative and quantitative). Multiple distinct c-semirings, each encoding a distinct dimension over which preference is specified (including mixes of qualitative and quantitative dimensions) can be combined in a simple fashion to obtain a single c-semiring [15] - thus providing a modular framework in which preference dimensions could be added or removed while leaving much of the reasoning machinery intact.

We start with the definition of a c-semiring [15].

Definition 1. [15]: A c-semiring is a 5-tuple $\langle A, \oplus, \otimes, \mathbf{0}, \mathbf{1}\rangle$ such that:

- A is a set of abstract preference values with $\mathbf{0}, \mathbf{1} \in A$ ($\mathbf{0}$ represents the "‘worst"’ preference value while $\mathbf{1}$ represents the "‘best"’ preference value);
- $\oplus$ is a binary operator which is closed (i.e. if $a, b \in A$, then $a \oplus b \in A$), commutative (i.e. $a \oplus b = b \oplus a$), associative (i.e. $a \oplus (b \oplus c) = (a \oplus b) \oplus c$), idempotent (i.e. if $a \in A$, then $a \oplus a = a$), has $\mathbf{0}$ as a unit element (i.e. $a \oplus \mathbf{0} = a = \mathbf{0} \oplus a$), and with $\mathbf{1}$ as an absorbing element (i.e. $a \oplus \mathbf{1} = \mathbf{1} = \mathbf{1} \oplus a$);
- $\otimes$ is a binary operator which is closed (i.e. if $a, b \in A$, then $a \otimes b \in A$), commutative (i.e. $a \otimes b = b \otimes a$), associative (i.e. $a \otimes (b \otimes c) = (a \otimes b) \otimes c$), has $\mathbf{1}$ as a unit element (i.e. $a \otimes \mathbf{1} = a = \mathbf{1} \otimes a$), and $\mathbf{0}$ as an absorbing element (i.e. $a \otimes \mathbf{0} = \mathbf{0} = \mathbf{0} \otimes a$);
- $\otimes$ distributes over $\oplus$ (i.e. $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$).

Intuitively, $\oplus$ is used to compare preference values, while $\otimes$ is used to combine preference values. The $\oplus$ operator generates a partial order $\preceq$ on A as follows: for any $v_1, v_2 \in A$, $v_1 \preceq v_2$ (read this as v_1 is "at least as good as" v_2) if $v_1 \oplus v_2 = v_1$.

In the constraint satisfaction literature, several useful instantiations of c-semirings have been discussed: boolean (where $A = \{T, F\}$), fuzzy (where $A = [0, 1]$), weighted (where $A = \mathcal{R}^+$) etc. Qualitative c-semirings can be of interest, where the $\otimes$ and $\oplus$ operators are defined extensionally. Consider the following c-semiring, where l represent *low*, m represents *medium* and h represents *high*: $Q = \langle \{l, m, h\}, \oplus, \otimes, l, h \rangle$ where $l \oplus m = m$, $m \oplus h = h$, $l \oplus h = h$, $l \otimes m = l$, $m \otimes h = m$, and $l \otimes h = l$. To rephrase this, this c-semiring allows us to use 3 preference values - *low*, *medium* and *high* - with *low* as the designated "worst" value (the element denoted by $\mathbf{0}$ in the c-semiring) and *high* as the designated "best" value (the element denoted by $\mathbf{1}$ in the c-semiring). If we were to compare these values using the $\oplus$ operator, the comparison of *low* and *medium* would yield *medium*, the comparison of *medium* and *high* would yield *high* and so on. Similarly, if we were to combine these values using the $\otimes$ operator, then the combination of *low* and *medium* would generate *low*, the combination of *medium* and *high* would generate *medium* and so on.

Notice that the set of preference values in each of the c-semiring instances discussed above offers a scale on which degree of compliance might be assessed. This is also true for c-semirings consisting only of abstract preference values, and for composite c-semirings obtained by combining several component c-semirings using the technique described in [15] (ommitted here for brevity). Thus, if we were using the qualitative c-semiring discussed above with preference values $\{l, m, h\}$, we obtain a vocabulary for describing degree of compliance that permits us to assert that a given state of affairs has a low (l) degree of compliance, or a medium (m) degree of compliance and so on. Similarly, one might conceive of a composite c-semiring consisting of this qualitative c-semiring combined with the fuzzy c-semiring discussed above which would allow us to assess preference on two separate dimensions using these two distinct scales, where degrees of compliance would be represented using pairs where the first element is a preference value from the qualitative c-semiring and the second element a value from the fuzzy c-semiring (e.g. $\langle l, 0.7 \rangle$ or $\langle m, 0.55 \rangle$).

In the following, we will take a state to be a complete assignment of values to a set of variables V (these do not necessarily have to be propositional, thus permitting us to also view solutions to constraint satisfaction or optimization problems as states). We define a *valued optimization norm* (so called because these norms associate states with specific preference and sanction values) as a mapping from a state to a semiring valuation of that state. In addition, we define a real-valued penalty associated with each state. We also constrain the penalties so that any penalty associated with a more preferred state has to be lower than a penalty associated with a less preferred state. If two states are equally preferred, then their penalties must be equal. Also, the penalty associated with the most preferred state must be 0.

Definition 2. *Given a c-semiring $P = \langle A, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$ and a set of states S (with penalties represented as elements of the set of reals $\mathcal{R}$), a* **valued optimization norm** *n_P is defined as $n_P : S \rightarrow A \times \mathcal{R}$, such that: (1) For any s such that $n_P(s) = \langle \mathbf{1}, r \rangle$, $r = 0$ and (2) For any s_1 and s_2, if $n_P(s_1) = \langle v_1, r_1 \rangle$ and $n_P(s_2) = \langle v_2, r_2 \rangle$ where $s_1, s_2 \in S$, $v_1, v_2 \in A$ and $r_1, r_2 \in \mathcal{R}$ and $v_1 \oplus v_2 = v_1$, then $r_1 \leq r_2$ (if $v_1 = v_2$, then $r_1 = r_2$).*

Example 1. Consider a simple example where our propositional vocabulary consists of 2 letters, p and q and we prefer the state $p \wedge q$ the most and the state

$\neg p \wedge \neg q$ the least. The states $\neg p \wedge q$ and $p \wedge \neg q$ are in-between most and least preferred and are equally preferred. A valued optimization norm n_Q conforming to the definition above could be defined as follows, using the c-semiring $Q = \langle \{l, m, h\}, \oplus, \otimes, l, h \rangle$ (together with its associated extensional definitions of $\oplus$ and $\otimes$) as in the discussion above: $n_Q(p \wedge q) = \langle h, 0 \rangle$, $n_Q(\neg p \wedge q) = \langle m, 5 \rangle$, $n_Q(p \wedge \neg q) = \langle m, 5 \rangle$, $n_Q(\neg p \wedge \neg q) = \langle l, 10 \rangle$.

Note that the definition above provides the basis for two important dimensions to practical compliance management. The first is *graded compliance.* Valued optimization norms permit us to associate an explicit *degree of compliance* value (effectively the corresponding c-semiring value) with each state of affairs. They also permit us to associate finer-grained sanctions (as opposed to a single sanction for the violation of a boolean norm) with different degrees of compliance.

It is easy to see how contrary-to-duty obligations can be represented in this formalization. Consider a reparation sequence (one way to represent a contrary-to-duty obligation) in FCL [16] ($\times$ is used in the following to mean "'else"') : $O_1 \times O_2 \times \ldots \times O_m$. The reparation sequence above is to be read as follows: obligation O_1 holds, failing which obligation O_2 holds and so on. Let $M(\phi)$ represent the set of states that satisfy ϕ (i.e., the models of ϕ). This reparation sequence can be represented by any norm n_P satisfying the following property: For any $s \in M(O_i)$ and any $s' \in M(O_j)$ where $i < j$ and $n_P(s) = \langle v, r \rangle$ and $n(s') = \langle v', r' \rangle$, $v \oplus v' = v$ and $r < r'$. Consider an FCL[16] reparation sequence: $(Op \wedge q) \times (O\neg p \wedge q) \times (O\neg p \wedge \neg q)$, where $O\phi$ represents the obligation to make ϕ true. n_Q defined in the example above is one instance of a valued optimization norm that encodes this reparation sequence. In general, a given reparation sequence could be encoded by a number of distinct valued optimization norms.

In the discussion in the subsequent sections, some of the development will be presented in terms of the abstract notion of optimization norms, without specifically referring to valued optimization norms. Some concepts will, however, be illustrated using valued optimization norms. The discussion on sanction management later in the paper relies entirely on our formalization of valued optimization norms.

4 Complying with Optimization Objectives

We first consider in abstract terms the machinery that we might use to select amongst alternative options. These options could be plans in BDI agents (often called *option selection* [17]) or intentions (the *intention selection* problem [17]) or, more generally, states that an agent might seek to realize. In general terms, the machinery we require must be able to make ordinal comparisons between states resulting from the optional courses of action that an agent might have to select from. For instance, given two states s_1 and s_2 and an optimization norm *maximize* x articulated in terms of a task measure x, the machinery must be able to make a determination on which of s_1 and s_2 lead to a higher value of x (without necessarily computing that value), or whether they lead to identical values of x.

As discussed in the introduction, one intuitive approach to establishing that an agent has complied with a optimization norm is to establish that its decisions sought to optimize the norm. This can be formalized in two ways:

– *Global compliance:* This requires that the final state achieved by an agent be the optimal (relative to the optimization norm) of all the states that were feasible for the agent to achieve. Note that this approach is only applicable when the agent performs bounded computation and a clear notion of final state exists. It also requires the ability to compute the set of all possible final states that would be feasible for the agent to achieve. It might also mean that agents would make local choices that not optimal with respect to the objective (in the interests of arriving at the globally optimal final state).
– *Local compliance:* This requires that every (local) choice made by the agent be optimal with respect to the objective. Note that this might mean that the final state arrived at by an agent (performing bounded computation) is sub-optimal. However, this notion of compliance can be used in agents performing unbounded computation.

5 Managing Norm Conflict

We address the question of norm conflict detection and resolution in this section. We begin by considering a well-known example of the interplay between norms and preferences.

Sen's Example. Sen [18] offers an example where the interplay between norms and objectives generates results that are contrary to what preference maximization would generate. Consider a situation where an agent prefers option x to y and y to z. Assume that the agent selects y from $\{x, y, z\}$ and selects z from $\{y, z\}$ (both choices run counter to the preference maximization principle). Sen offers an account involving norms that explains such behaviour. Assume that the options are differently-sized slices of a cake, and that x is the biggest slice, y the next in size and z the smallest. The agent has a preference for larger slices of the cake. To explain the behaviour of the agent, Sen brings to bear a "politeness" norm, which requires the agent to not select the largest slice when asked to choose from a set of cake slices. In selecting amongst $\{x, y, z\}$, the agent uses this norm to rule out option x and then selects the preferred option from the remainder (i.e., option y). Similarly, in selecting amongst $\{y, z\}$, the agent rules out y as an "impolite" choice (it is the largest of the 2 choices) and selects the remaining option (z).

We argue that viewing the politeness norm as a preference ordering (which prefers any state where a non-largest slice of cake is selected over a state where the largest slice is selected), together with a prioritization on objectives (where the optimization norm derived from the politeness norm has priority over the optimization norm to maximize the size of the cake slice selected), offers an equally valid explanation of the agent's behaviour. Indeed, our approach supports finer grained reasoning, for instance, by permitting us to explore the pareto-frontier with respect to these two objectives (if they were treated as having equal priority).

We need, in the first place, a formal definition of conflict for optimization norms (with other optimization norms as well as with boolean norms). We will define consistency for optimization norms by viewing them as preference relations on the set of feasible solutions (recall the discussion of these issues in the introduction). We can then define three distinct notions of consistency for optimization norms.

- *Absolute consistency:* Under this notion, a pair of optimization norms o_1 and o_2 are deemed to be inconsistent if and only if there exists a pair of solutions (or options) s and s' such that s is preferred over s' under o_1 and s' is preferred over s under o_2. Absolute consistency is arguably an overly stringent notion. Our intent here is to identify situations where objectives might "pull in different directions"'. Under this notion of consistency, we would deem a pair of optimization norms to be inconsistent even if the pair s and s' whose relative ordering they disagreed on were actually not in the set of options currently being deliberated on. In other words, we would deem the objectives to be inconsistent even if they were not "pulling in different directions" in the current instance. Note also that checking consistency in this mode requires that we extensionally elaborate the preference relation corresponding to the optimization norm with reference to the set of all possible options over which preferences might be specified - but that set is potentially unbounded and cannot in general be predicted. Checking for absolute inconsistency is therefore impractical.
- *Contextual consistency:* Under this notion, a pair of optimization norms o_1 and o_2 are deemed to be inconsistent in a given context C (defined as a set of options/solutions from which a choice has to be made) if and only if there exists a pair of solutions (or options) $s, s' \in C$ such that s is preferred over s' under o_1 and s' is preferred over s under o_2. The notion of contextual consistency helps us determine whether a pair of objectives would lead to conflicting preferences in a particular context, given a particular set of alternatives. It could be argued that this too is a somewhat stringent notion, since conflicting preferences might not manifest themselves in actual conflicting choices if the conflicting preferences involve options that are not the top choices (i.e., the most preferred options) under the two preference orderings. In other words, an agent could "tolerate" optimization norms which are deemed to be both absolutely inconsistent and contextually inconsistent, as long as this does not lead to conflicting choices. On the other hand, if the context changes infrequently, this notion of consistency can be useful (if the relevant optimization norms are - or can be made - consistent for that context, no further inconsistency handling will be required while that context remains unchanged). Checking for consistency over a set of optimization norms involves a straightforward generalization of the pair-wise check mentioned above.

– *Choice consistency:* Under this notion, a pair of optimization norms o_1 and o_2 are deemed to be inconsistent in a given context C (defined as a set of options/solutions from which a choice has to be made) if and only if S is the set of strictly non-dominated solutions (or options) under o_1, S' is the set of strictly non-dominated solutions (or options) under o_2, $S \subseteq C$, $S' \subseteq C$ and $S \cap S' = \emptyset$. Here, the set of strictly non-dominated solutions consists of those solutions for which there exists no other that is strictly more preferred under the given ordering.

We now consider the question of conflict between optimization norms and boolean norms. Optimization norms and boolean norms do not conflict in an absolute sense, i.e., independent of a given context (given by set of available alternatives). To understand why, we need to recall a commonly used definition of inconsistency: two assertions are inconsistent if there does not exist a model which satisfies both. Note that the notion of a model "satisfying"' an optimization objective is undefined. Viewed in terms of preferences over models or states, the notion of compliance with an optimization norm that we have defined above requires not just the preferred state/model but all of the other states/models that were available as options before we can determine compliance with an optimization norm. We cannot therefore speak of the inconsistency of an optimization norm with a boolean norm, independent of richer contextual information, in any meaningful sense.

However, an optimization norm o will be deemed to be inconsistent with a boolean norm n in a given context C if and only if every element of the set S (where $S \subseteq C$) of strictly non-dominated (most preferred) options according to o violates n. We assume, as before, that the options in question are actions/tasks, or plans or states of affairs, so that in each case we are able to check for the violation of boolean norms. Note that we can only define inconsistency using intuitions similar to those for *choice consistency* for optimization norms.

We will now consider an example that illustrates these notions of inconsistency, using valued optimization norms. Note that the focus in the example is on the c-semiring valuations associated with states of affair - the associated penalties/sanctions do not play a role in the example (but become critical in the later discussion on sanction management).

Example 2. Recall, from Example 1, the valued optimization norm n_Q using the c-semiring $Q = \langle \{l, m, h\}, \oplus, \otimes, l, h \rangle$ (together with its associated extensional definitions of $\oplus$ and $\otimes$): $n_Q(p \wedge q) = \langle h, 0 \rangle$, $n_Q(\neg p \wedge q) = \langle m, 5 \rangle$, $n_Q(p \wedge \neg q) = \langle m, 5 \rangle$, $n_Q(\neg p \wedge \neg q) = \langle l, 10 \rangle$. Consider another valued optimization norm $n2_Q$ defined (using the same c-semiring Q) as: $n2_Q(p \wedge q) = \langle h, 0 \rangle$, $n2_Q(\neg p \wedge q) = \langle m, 5 \rangle$, $n2_Q(\neg p \wedge \neg q) = \langle m, 5 \rangle$, $n2_Q(p \wedge \neg q) = \langle l, 10 \rangle$. Under the notion of absolute inconsistency, n_Q and $n2_Q$ are inconsistent, since the state $p \wedge \neg q$ is preferred over $\neg p \wedge \neg q$ under n_Q while the opposite preference holds under $n2_Q$. If the current context is defined by these two states, then this also illustrates contextual inconsistency. Consider a third valued optimization norm $n3_Q$ defined on the same c-semiring Q: $n_Q(p \wedge \neg q) = \langle h, 0 \rangle$, $n_Q(p \wedge q) = \langle m, 5 \rangle$, $n_Q(\neg p \wedge \neg q) = \langle m, 5 \rangle$, $n_Q(p \wedge q) = \langle l, 10 \rangle$. If the context is defined by states satisfying $p \wedge q$

and $p \wedge \neg q$, then the norms n_Q and $n3_Q$ are inconsistent under the notion of choice consistency (the strictly non-dominated states under the two norms lead to incosistent states). In the same context, the boolean norm $p \wedge \neg q$ contradicts the valued optimization norm n_Q (note that the set of strictly non-dominated states under n_Q consists of a single state: $p \wedge q$).

The resolution of inconsistency in any theory leads to changing the theory in some way. The logic of the theory change [19] argues that when we are obliged to make changes to a theory, we should try to minimize the extent of change, given that theories encode knowledge or intent or deontic constraints which we should alter as little as possible while still accommodating the change that needs to be implemented. Arguably, the same intuitions apply when we resolve inconsistencies amongst norms.

The three notions of consistency discussed above lead to three corresponding notions of resolution:

Resolving absolute inconsistency: Resolving this type of inconsistency requires that an assertion of the form $s' \prec s$ (*s is preferred to* s') be removed from one of the pair of optimization norms that are found to be inconsistent. As discussed before, this notion of consistency is in general of little practical value, except in settings where the set of all alternatives that an agent might ever have to select amongst is known a priori (in which case it effectively reduces to contextual consistency).

Resolving contextual inconsistency: This involves the same machinery as in the case of absolute inconsistency. Observe that inconsistency resolution means that one or more of the currently applicable set of optimization norms is relaxed. In the event that inconsistency is detected over a set of optimization norms $O = \{o_1, o_2, \ldots, o_m\}$ where $O_s \subseteq O$ consists of optimization norms which prefer s over s' and $O_{s'} \subseteq O$ is the set of optimization norms which prefer s' over s, then we might use majority as the basis for deciding which of the preferences get relaxed. Thus, if $|O_{s'}| < |O_s|$, we might choose to remove $s \prec s'$ from each element of $O_{s'}$. In other settings, the criteria to determine which optimization norm to relax would be domain-dependent (in mixed-initiative reasoning settings, one might even ask the user for guidance on this). Observe that once inconsistency has been resolved, we could take the union of the resulting set of preference relations to guide choice. A range of other intuitions are explored in computational social choice theory (see [20] for a survey).

Resolving choice inconsistency: In the case of a set of optimization norms generating sets of strictly non-dominated (most preferred) options that do not intersect, we would have to pick the "winning" set of norms (whose top choices would determine an agent's selections). As with contextual inconsistency, we could use majority as the basis for deciding the winners - alternatively, the criteria would be domain-dependent.

In the account of inconsistency resolution above, we had to rely on a representation of an optimization norm in the form of a preference relation. We can explore the resolution of inconsistency between optimization norms using two additional intuitions, which do not require this reduction to a preference relation:

Pareto-optimal solutions: The notion of pareto-optimality, frequently used in decision theory, provides an alternative basis for resolving inconsistency amongst optimization norms. In the following, we will use $o(s)$ to denote the value of the objective o for option s. Given a set of optimization norms $O = \{o_1, o_2, \ldots, o_n\}$, which are viewed uniformly, and without loss of generality, as maximization objectives and a set of options $S = \{s_1, s_2, \ldots, s_m\}$, a *pareto-optimal solution* is typically defined as some $s_i \in S$ such that there exists no $s_j \in S$ for which $o_k(s_j) > o_k(s_i)$, for at least one $o_k \in O$, and for all other $o_i \in O$ where $i \neq k$, $o_i(s_j) \geq o_i(s_i)$. In other words, a pareto-optimal option is one for which there exists no other feasible option which performs strictly better on one objective and at least as well on all of the others. The term *pareto-frontier* is often used to describe the set of all pareto-optimal solutions. In our setting, the pareto-frontier can be viewed as consisting of alternative resolutions of optimization norm inconsistency. These can be presented as choices to users in a mixed-initiative reasoning setting (the system only filters, but does not make the final choice in such settings).

Prioritization of objectives: A well-known approach to dealing with multiple objectives in operations research is to create a weighted sum of these objectives, with the weights reflecting the relative priorities of the corresponding objectives. There is a critical assumption here that the objective functions map options to commensurate scales (what would happen if one objective measured cost in dollars and another measured time in seconds?). If all of the objectives are equally weighted, then each solution to the resulting optimization problem represents an element of the pareto-frontier. In our alternative account of Sen's cake-choice example, let o_{NL} represent the optimization norm that makes us prefer states where we haven't eaten the largest slice of cake to states where we have. Let o_L represent our preference for eating larger slices of a cake. We simplify our discussion by avoiding the formalization of the commensurate scales on which the two objectives would evaluate options. The weighted combination of the two objectives would be of the form $w_{NL}.o_{NL} + w_L.o_L$ where $w_{NL} > w_L$ (for the purposes of our example, the actual weights are immaterial as long as this inequality holds).

We now need to address the question of resolving inconsistencies between optimization norms and boolean norms. The solution here is fairly simple: optimization norms can be relaxed while boolean norms cannot. Recall that an optimization norm o will be deemed to be inconsistent with a boolean norm n in a given context C if and only if every element of the set S (where $S \subseteq C$) of strictly non-dominated (most preferred) options violates n. This inconsistency can be resolved if we are able to promote at least one n-satisfying state (say s) to become a member of S. Given our earlier discussion on the need to minimize change to the original specification of a norm, we could explore several intuitions on the specific changes required in the underlying preference relation (as before, viewed as a set of preference assertions). The general approach would be to identify an o' which satisfies the following conditions: (1) There exists at least one n-satisfying element of the set of strictly non-dominated options under o', (2) There exists no o'' where $(o \Delta o'') \subseteq (o \Delta o')$ which also satisfies condition (1) (here Δ refers to the symmetric set difference operator).

An important question to address is the distinction between the *violation* and *relaxation* of optimization norms. We *relax* an optimization norm to ensure consistency with other norms. The notion of norm compliance discussed earlier continues to provide a clear yardstick for deciding whether an optimization norm (or a set of such norms) has been violated.

6 Sanction Management

We now consider the problem of norm compliance in multi-agent systems, and in particular the problem of how graded sanctions (corresponding to graded degrees of non-compliance) might be *decomposed* and assigned to the (possibly many) agents responsible for a (graded) violation. Social norms may need to be decomposed to obtain agent-specific obligations in a multi-agent context. As discussed earlier, this is a problem of practical importance whenever agents delegate responsibility to other agents. In a more general business setting, a formal understanding of the decomposition of sanctions is critical in managing outsourcing and in formalizing the relationship between a contract and sub-contracts whenever sub-contracting is involved. A formalization is complicated by the fact that the sanctions associated with different levels of compliance/violation are contextually determined, specifically by the number of agents involved in satisfying the norm in question, and the level to which each of these comply with or violate the norm.

In the following, given a set of agents Ag and a set of variables Var, we will use the function $\theta : Ag \rightarrow 2^{Var}$ to map an agent to a set of variables that the agent is responsible for. Also in the following, given a norm n_P which specifies preferences using an underlying semiring P, we will use $bestval_{n_P}(ag)$, to denote the set of preference values assigned to the set of value assignments (states) that include the current assignments of values to variables in $\theta(ag)$ by n_P such that there exists no state s that also includes the current assignment of values to $\theta(ag)$ where $n_P(s) = \langle v', r' \rangle$ and $v' \prec v$ for some $v \in bestval_{n_P}(ag)$ (here $\prec$ is the strict version of the partial order $\preceq$ associated with the c-semiring P).

Definition 3. *Given a multi-agent norm context $\langle n_P, Var, Ag, \theta \rangle$ where n_P is the norm in question, Var is the set of variables over which the states that the norm refers to are defined, Ag is the set of agents jointly responsible for satisfying n_P and the $\otimes$ operator associated with the c-semiring P is idempotent, and given a complete assignment s of values to variables in V where $n_P(s) = \langle v, r \rangle$, the (real-valued) sanction applied on an agent ag_i is defined as:*

- *If $bestval_{n_P}(ag_i) = v$ then the sanction incurred by ag_i is r/m where $Ag_X = \{ag \mid bestval_{n_P}(ag) = v\}$ and $|Ag_X| = m$.*
- *If $bestval_{n_P}(ag_i) \neq v$ then sanction incurred by ag_i is 0.*

The idempotence property of the $\otimes$ operator, which states that $a \otimes a = a$ for any preference value a is key to understanding the definition above. Operators such as *min* and *max* are idempotent, while the arithmetic addition operator $+$ is not. With an idempotent combination operator such as *min*, the contribution of an agent to the final preference value is only of interest if that final preference value equals the best value assigned by the valued optimization norm to the portion of the state determined by that agent. If the best value happens to be higher, then the agent in question was not in fact responsible for the final value of that state (some other agent or agents whose component of the final state was evaluated to the eventual valuation of the state would have been responsible instead). Similarly, if the contribution of the agent had a lower valuation, then the whole state would have had that lower valuation, hence that case is not of interest either.

As an example, consider a compliance requirement that both p and q have to be completely satisfied. Assume that the c-semiring being used is Q as defined above and that both p and q can be satisfied at one of the following 3 levels: COMPLETELY (represented by the c-semiring value h), PARTIALLY (represented by m), NOT-AT-ALL (represented by l). Consider 2 agents: A_1 and A_2. A_1 is responsible for satisfying p. $A_2$is responsible for satisfying q. The compliance requirement is formalized by the valued optimization norm $n1_Q$ as follows: $n1_Q(\langle p = h, q = h\rangle) = (h, 0)$, $n1_Q(\langle p = h, q = m\rangle) = (m, 50)$, $n1_Q(\langle p = m, q = h\rangle) = (m, 50)$, $n1_Q(\langle p = m, q = m\rangle) = (m, 50)$, $n1_Q(\langle p = l, q = l\rangle) = (l, 100)$, $n1_Q(\langle p = h, q = l\rangle) = (l, 100)$ (note that we need a total of 9 such assertions - we do not list them all for brevity). Notice that the preference value assigned to each state is obtained by applying the $\otimes$ operator on the preference values associated with p and q.

– Scenario 1: The requirement is violated completely (incurring a penalty of 100) because A_2 fails to satisfy q completely, i.e. as a NOT-AT-ALL (even though A_1 satisfies p completely). Here A_2 pays a penalty of 100.
– Scenario 2: The requirement is violated partially (incurring a penalty of 50) because A_2 fails to satisfy q partially (even though A_1 satisfies p completely). Here A_1 pays a penalty of 50.
– Scenario 3: The requirement is violated partially (incurring a penalty of 50) because both A_1 and A_2 satisfy q and p (respectively) partially. Here A_1 and A_2 pay a penalty of 25 each

We do not list all of the scenarios here for brevity. The formalization above does not cover the complete sanction decomposition problem, but offers pointers on how the full problem might be solved using an approach like this. In the non-idempotent case, if the $\otimes$ operator performs arithmetic addition of cost, it is easy to decompose penalties to agents in direct proportion to their contribution to costs. In other cases, the complexity might arise due to evaluating the contribution of each agent to an eventual outcome.

7 Related Implementions of Optimization in Agent Deliberation

It is useful, at this point, to consider the implementation of optimization norms in agent systems. The theory of compliance for optimization norms discussed in this paper has been implemented in the CASO BDI agent programming language [21], which provides pointers on how optimization objectives might be integrated into agent deliberation (but it does not support the framework for inconsistency detection and resolution developed in this paper). The key idea is that CASO agents accept optimization objectives as events in their event queues. A CASO agent uses these objectives in option selection (choosing between competing plans)and intention selection. In the absence of a clear notion of termination (common to most BDI agent implementations), a CASO agent cannot look ahead to a final state (as required by the notion of global compliance defined in this paper). Instead, a CASO agent achieves local compliance by pruning the goal-plan tree obtained from its current options at a parametric depth, and then exploring all paths to the pseudo-leaf nodes obtained. In deciding which option to commit to, a CASO agent solves an optimization problem, using the currently applicable optimization objectives, once for each option (CASO plan contexts contain both constrain and non-constraint predicates, leading to a representation akin to constraint logic programming). A CASO agent then selects the option that offers the optimal value with respect to the currently applicable optimization norms. Intention selection in CASO uses similar machinery.

In the BAOP agent programming language [22], both objective functions and (c-semiring) valued preferences are brought to bear on agent deliberation. The c-semiring preferences and specified with respect to domain states. The traditional AgentSpeak-style plans of CASO are therefore annotated with effects, and machinery defined to propagate effects over plans and sub-plans. As with CASO, BAOP provides pointers to how optimization norms might be integrated into agent deliberation, but does not make provision for inconsistency detection and resolution.

The rest of the machinery described in this paper has not been implemented. However, as the discussion above suggests, much of the conceptual framework presented should be possible to implement in a straightforward manner. On the one hand, the need to solve optimization problems during agent deliberation adds to its complexity. On the other hand, this framework provides tantalizing hints on how agent programming environments could be based entirely on (efficient) optimization technology.

8 Conclusions and Future Work

We have argued in this paper that optimization norms represent an important dimension to normative reasoning in multi-agent systems. We have provided a conceptual framework to support reasoning with optimization norms, and extended it to provide support for graded compliance, graded sanctions and sanction decomposition. The observations provided in this paper provide a starting point for an interesting strand of research. Ultimately, this raises the question: are the problems of compliance checking and non-compliance resolution instances of the more general optimization problem?

References

1. Boer, A., van Engers, T., Winkels, R.: Mixing legal and non-legal norms. In: Proceedings of the 2005 Conference on Legal Knowledge and Information Systems: JURIX 2005: The Eighteenth Annual Conference, Amsterdam, The Netherlands, pp. 25–36. IOS Press (2005)
2. van der Torre, L., Tan, Y.H.: Contrary-to-duty reasoning with preference-based dyadic obligations. Annals of Mathematics and AI 27, 49–78 (1989)
3. Frank Dignum, D.K., Sonenberg, L.: From desires, obligations and norms to goals. Cognitive Science Quarterly 2, 407–430 (2002)
4. Panagiotidi, S., Vazquez-Salceda, J.: Norm-aware planning: Semantics and implementation. In: 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 3, pp. 33–36 (2011)
5. Mares, E.D., MacNamara, P.: Supererogation in deontic logic: Metatheory for dwe and some close neighbours. Studia Logica 57, 397–415 (1997)
6. Simon, H.A.: Models of bounded rationality. MIT Press, Cambridge (1982)
7. Morrison, E., Ghose, A.K., Koliadis, G.: Dealing with imprecise compliance requirements. In: Proceedings of the 2nd International Workshop on Dynamic and Declarative Business Processes (DDBP 2009). IEEE Computer Society Press (2009)
8. Koliadis, G., Desai, N., Narendra, N.C., Ghose, A.K.: Analyst-mediated contextualization of regulatory policies. In: Proceedings of the IEEE International Services Computing Conference (IEEE SCC 2010), Miami, USA, pp. 281–288. IEEE Computer Society Press (July 2010)
9. Governatori, G.: Representing business contracts in ruleml. International Journal of Cooperative Information Systems 14, 181–216 (2005)
10. Udupi, Y.B., Singh, M.P.: Contract enactment in virtual organizations: A commitment-based approach. In: Proceedings of the 21st National Conference on Artificial Intelligence, AAAI (July 2006)
11. Villatoro, D., Andrighetto, G., Sabater-Mir, J., Conte, R.: Dynamic sanctioning for robust and cost-efficient norm compliance. In: IJCAI, pp. 414–419 (2011)
12. Boella, G., van der Torre, L.W.N.: Negotiating the distribution of obligations with sanctions among autonomous agents. In: Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI 2004, pp. 13–17 (2004)
13. von Wright, G.H.: Deontic logic. Mind 60(237), 1–15 (1951)
14. Elster, J.: Fairness and norms. Social Research: An International Quarterly 73(2), 365–376 (2006)
15. Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., Fargier, H.: Semiring-based csps and valued csps: Frameworks, properties, and comparison. Constraints 4(3), 199–240 (1999)
16. Governatori, G., Milosevic, Z.: An approach for validating bcl contract specifications. In: 2nd EDOC Workshop on Contract Architectures and Languages (CoALA 2005). IEEE Digital Library (2005)
17. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Perram, J.W., Van de Velde, W. (eds.) MAAMAW 1996. LNCS, vol. 1038, Springer, Heidelberg (1996)

18. Sen, A.K.: Internal consistency of choice. Econometrica 61, 495–521 (1993)
19. Alchourron, C., Gardednfors, P., Makinson, D.: On the logic of theory change: Partial meet functions for contraction and revision. Journal of Symbolic Logic 50, 510–530 (1985)
20. Chevaleyre, Y., Endriss, U., Lang, J., Maudet, N.: A Short Introduction to Computational Social Choice. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 51–69. Springer, Heidelberg (2007)
21. Dasgupta, A., Ghose, A.K.: Implementing reactive bdi agents with user-given constraints and objectives. International Journal of Agent-Oriented Software Engineering 4(2), 141–154 (2010)
22. Dasgupta, A., Ghose, A.K.: BDI Agents with Objectives and Preferences. In: Omicini, A., Sardina, S., Vasconcelos, W. (eds.) DALT 2010. LNCS, vol. 6619, pp. 22–39. Springer, Heidelberg (2011)

Norm Emergence through Dynamic Policy Adaptation in Scale Free Networks

Samhar Mahmoud[1], Nathan Griffiths[2], Jeroen Keppens[1], and Michael Luck[1]

[1] Department of Informatics, King's College London, London WC2R 2LS, UK
samhar.mahmoud@kcl.ac.uk
[2] Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

Abstract. As has been stated elsewhere, norms are a valuable means of establishing coherent cooperative behaviour in decentralised systems in which there is no central authority. Axelrod's seminal model of norm establishment in populations of self-interested individuals provides some insight into the mechanisms needed to support this through the use of *metanorms*, but considers only limited scenarios and domains. While further developments of Axelrod's model have addressed some of the limitations, in particular in considering its application to different topological structures, this too has been limited in not offering an effective means of bringing about norm compliance in scale-free networks, due to the problematic effects of *hubs*. This paper offers a solution, first by adjusting the model to more appropriately reflect the characteristics of the problem, and second by offering a new dynamic policy adaptation approach to learning the right behaviour. Experimental results demonstrate that this dynamic policy adaptation overcomes the difficulties posed by asymmetric distribution of links in scale-free networks, leading to an absence of norm violation, and instead norm emergence.

1 Introduction

Norms are an effective means of governing the behaviours of different members of decentralised open systems, such as P2P file-sharing systems in which cooperation between members maintains benefits for all. However, individuals often take benefits without contributing to the common good, the *free riding* phenomenon [1] by which some download files from others without uploading in return. In decentralised systems, the absence of a central authority means that there is no consequence for such behaviours. Many researchers ([4,6,7,12,14,16]) have proposed norms as a means of regulating agent behaviour but, as shown by Axelrod [2], norms alone may not lead to desired outcomes. In consequence, Axelrod proposed *metanorms* as a means of ensuring not that norms are complied with, but that they are enforced. He showed that metanorms are effective in fully-connected networks, but did not consider other kinds of topology.

Some work has already been undertaken on examining the impact of different topologies on norm establishment. For example, Savarimuthu et al. [11] consider the *ultimatum game* in the context of a role model that provides advice on whether to change norms in order to enhance performance, and provide experimental results for random and scale-free networks. Delgado et al. [5] study norm emergence in coordination games in scale-free networks, and Sen et al. [13] similarly examine rings and scale-free networks in a related context. Additionally, Villatoro et al. [15] explore norm emergence within lattices and scale-free networks. While these efforts provide valuable and useful results, the context of application has tended to be limited, with only two agents involved in a single interaction, rather than a larger population. This simplifies the problem compared to those in which *multiple* agents are involved in a single interaction can impact on norm establishment.

Rather than adopting a fundamentally different model, in this paper we examine the problem of norm establishment in Axelrod's original model but extended to address the issues arising in topological structures, and in particular scale-free networks, which cause two significant problems. First, Axelrod's model assumes a fully connected network, and is predicated on that for certain aspects, such as how one agent observes another's actions. In a variably connected structure, this part of the model is thus not meaningful and requires modification, causing some difficulties in establishing norms. Second, in scale-free networks, which contain both heavily connected nodes (*hubs*) and lightly-connected nodes (*outliers*), hubs strongly influence norm emergence since they are involved in observation of, and interaction with, so many others in the network. While the work of Galan et al. [8] addresses the first point, applying Axelrod's model to other networks, the approach requires inappropriate access to the strategy of others [10].

In response, this paper provides two key contributions: it addresses a weakness in a previous technique for lattices and small-worlds to be consistent with the requirements of agent autonomy, and it provides a dynamic policy adaptation mechanism that leads to norm emergence in scale-free networks for which prior efforts have not succeeded. The paper begins with a brief description of the *metanorm model*. Section 3 then considers the problems that arise from the use of scale-free networks, and the adaptation of the model to cope with their characteristics. Section 4 introduces our solution for achieving norm emergence in this context and, finally, Section 5 concludes.

2 The Metanorms Model

Inspired by Axelrod's model [2], our simulation focusses only on the essential features of the problem. In the simulation, the agents play a game iteratively; in each iteration, they make a number of binary decisions. First, each agent decides whether to comply with the norm or to defect. Defection brings a reward for the defecting agent, and a penalty to all other agents, but each defector risks being observed by the other agents and punished as a result. These other agents thus decide whether to punish agents that were observed defecting, with a low penalty for the punisher and a high penalty for the punished agent. Agents that do not punish those observed defecting risk being observed themselves, and potentially incur metapunishment. Thus, finally, each agent decides whether to metapunish agents observed to spare defecting agents. Again, metapunishment comes at a high penalty for the punished agent and a low penalty for the punisher.

The behaviour of agents in each round of the game is random, but governed by three variables: the probability of being seen, *boldness*, and *vengefulness*. In each round, agents have a fixed number of opportunities to defect, each of which has a randomly selected probability of a defection being seen. Then, if an agent's *boldness* exceeds the

Algorithm 1. The Simulation Loop

```
1. for each round do
2.     interact()
3.     learn()
```

probability of a defection being seen, the agent defects. *Vengefulness* is the probability that an agent punishes or metapunishes another agent. Thus, the boldness and vengefulness of an agent are said to comprise that agent's *strategy* (or *policy* which we use when referring more specifically to agent actions). After several rounds of the game, each agent's rewards and penalties are tallied, and successful and unsuccessful strategies are identified. By comparing themselves to other agents on this basis, the strategies of poorly performing agents are revised such that features of successful strategies are more likely to be retained than those of unsuccessful ones. While successful strategies for agents increase their scores, our aim, in contrast, is to find strategies that will establish norms in the community: i.e. where agents tend to abide by the norm (due to low boldness) and tend to enforce and meta-enforce the norm (due to high vengefulness).

2.1 Evolution vs. Learning

Axelrod's evolutionary approach works well as a model to identify the *fittest* strategies because all agents act in an identical context. With the introduction of a topology, each agent possesses a potentially unique context defined by the agents to which it is connected. As such, strategies are not freely interchangeable. An agent typically keeps its strategy private, but adapts it in response to perceived effectiveness. Mahmoud et al. addressed these concerns by means of a reinforcement learning mechanism [10]; since we base our work on this starting point, we briefly describe those algorithms before presenting our own contributions. As illustrated in Algorithm 1, the simulation consists of several rounds, comprising an *interaction phase* and a *learning phase*, both of which rely on the identification of different components in the makeup of the utility (or *score*), which we introduce first. In order to assess and improve performance, each agent has a measure of utility, represented by different *scores*: defection score (DS), punishment score (PS), punishment omission score (POS) and total score (TS). DS is the utility gained or lost from defecting, PS is the utility lost from punishing others, and POS is the utility lost from not punishing defections. Finally, TS is the sum of these scores, in addition to the loss of utility that is borne by agents as a result of others defecting.

The interaction phase is concerned with the interaction between agents. As shown in Algorithm 2, each agent has various defections opportunities (o), in which it decides to defect (to violate the norm) if its boldness is greater than the probability of its defection being seen (S_o). As a result, the agent's DS is increased by a temptation value T, while every neighbouring agent NB (all agents that are directly connected) is hurt, reflected through a negative value H being added to the agent's total score. However, each hurt agent can observe the defection and react to it by imposing a punishment with probability corresponding to the observing agent's vengefulness. If an agent is punished, then its DS is decreased by a value P, and the punishing agent's PS is decreased with enforcement cost E. If an observing agent does not punish then, in

Algorithm 2. interact()

```
1.  for each agent $i$ do
2.    for each opportunity to defect $o$ do
3.      if $B_i > S_o$ then
4.        $DS_i = DS_i + T$
5.        for each agent $j \in NB_i$: $j \neq i$ do
6.          $TS_j = TS_j + H$
7.          if see($j$,$i$,$S_o$) then {$j$ sees $i$}
8.            if punish ($j, i, V_j$) then {$j$ punishes $i$}
9.              $DS_i = DS_i + P$
10.             $PS_j = PS_j + E$
11.           else
12.             for each agent $k \in NB_j : k \neq i \wedge k \neq j$ do
13.               if see($k$,$j$,$S_o$) then
14.                 if punish ($k, j, V_j$) then
15.                   $PS_k = PS_k + E$
16.                   $POS_j = POS_j + P$
```

turn, neighbours that observe this can metapunish the agent, again with probability corresponding to vengefulness. This results in the metapunished agent's POS being decreased by P (and thus increased in magnitude), since the metapunishment is a result of not punishing the defector, while the metapunishing agent's PS is reduced by E.

In the learning phase (Algorithm 3), the various scores are used as a means of improving performance in each round. Agents change their policies for action in the direction that should result in better scores. Initially, TS is calculated by accumulating the various component scores, and this is then used to determine whether to modify its policy, by comparing TS with the average population score (since agents that perform well should not change). If an agent's defection score DS is positive then it increases boldness, and decreases it if negative. Conversely, vengefulness is increased if PS is better than POS, and decreases otherwise. These changes arise by adding or subtracting a learning rate δ. Moreover, to explore the policy space, an agent may completely change its boldness and vengefulness values, determined by an *exploration rate*, γ.

2.2 Metanorms, Lattices and Small Worlds

As Mahmoud et al. [9] demonstrate, applying this model to fully-connected networks, lattices and small worlds results in norm emergence with different levels of success corresponding to the characteristics of the topologies. More specifically, the model always results in a population of agents with low average boldness and varying degrees of high average vengefulness. However, both lattices and small worlds have the attribute that *neighbourhood size* determines the number of neighbours to which each agent must be connected, and this appears to be important for convergence to norm emergence, with larger neighbourhoods giving better vengefulness. Conversely, *population size* has no effect on lattices, but in small worlds a larger population decreases vengefulness.

Algorithm 3. learn()

```
1.  for each agent $i$ do
2.      $TS_i = TS_i + DS_i + PS_i + POS_i$
3.      if $TS_i < AvgS_{NB_i}$ then
4.          if explore($\gamma$) then
5.              $B_i = random()$
6.              $V_i = random()$
7.          else
8.              if $DS_i < 0$ then
9.                  $B_i = B_i - \delta$
10.             else
11.                 $B_i = B_i + \delta$
12.             if $PS_i < POS_i$ then
13.                 $V_i = V_i - \delta$
14.             else
15.                 $V_i = V_i + \delta$
```

3 Scale-Free Networks

The topologies considered above are similar in that each agent has exactly the same number of connections, in contrast to scale-free networks [3], in which connections between nodes follow a power law distribution. Thus, some nodes have a vast number of connections, but the majority have very few connections, as illustrated in Figure 3. These properties of scale-free networks suggest an imbalance in connections. In turn, this has an impact on the results that can be obtained, due both to punishment and to enforcement costs, which dramatically modify the dynamics of the system. To investigate this, we ran 1000 experiments on a scale-free network with 1000 agents, five of which were *hubs* (having a large number of connections) and the others (which we call *outliers*) with at least two connections to other agents in the population, and typically no more than four connections (according to Barabasi's algorithm [3]). Each experiment was run for 1000 rounds (or timesteps), and parameters for the experiments were as

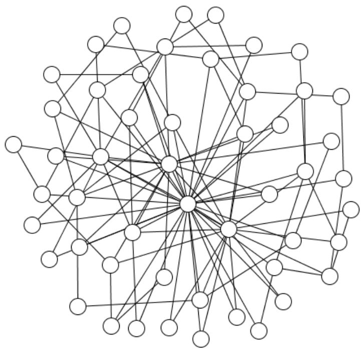


Fig. 1. Example of a Scale-free network

follows (and are the same for all subsequent experiments reported in this paper): $T = 3$, $E = -2$, $P = -9$, $H = -1$, $\delta = \frac{1}{7}$ and $\gamma = 0.01$. The results, shown in Figure 3, indicate that all runs end with both average boldness and average vengefulness, so that no norm is established. However, a detailed analysis of individual runs reveals that this is because there is no significant change to the average vengefulness and boldness, with both fluctuating around the average from the start of the run until the end.

By differentiating between hubs and outliers, some patterns are revealed, however. In particular, the model succeeds in lowering the boldness of hubs, but their vengefulness remains near average. Because hubs are connected to many other agents and are punished many times for a defection, boldness decreases. Conversely, they also punish many of these other agents for defecting, and consequently pay a very high cumulative enforcement cost that causes them to lower their vengefulness. In turn, this lower vengefulness causes them subsequently not to punish others and as a result to receive metapunishment from other hubs, leading to an increase in vengefulness again. Over time, this repeats, with vengefulness decreasing and then increasing back to the average, as shown in Figure 3(a) and 3(c). Note that for all results that are provided in this paper, we show both the long terms results of 1000 timesteps for completeness and the short term results of 100 timesteps for clarity.

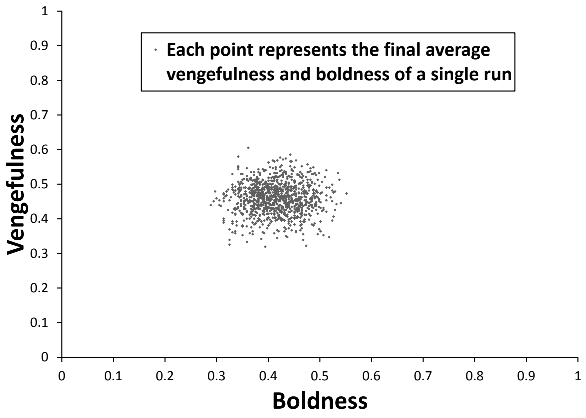
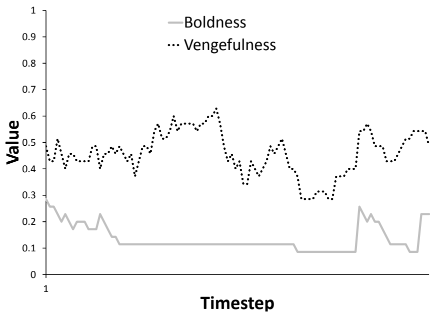
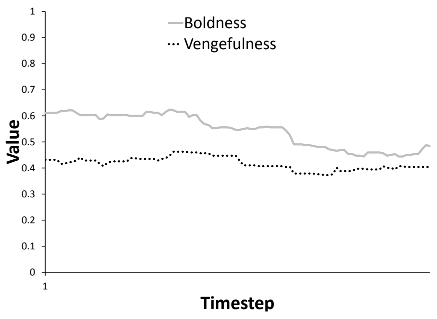


Fig. 2. Overall Result - 1000 Runs, 1000 Agents, 1000 Timesteps

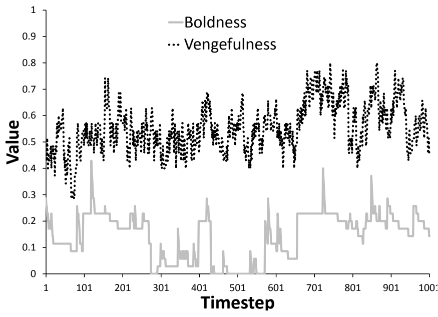
For the remaining, *outlier*, agents, changes to boldness and vengefulness are indicative of overall boldness and vengefulness because they comprise the majority of the population. They are typically connected to one or more of the hubs, and while they too defect and punish, they do so much less frequently than the hubs to which they are connected. Thus, their scores are higher than the scores of the hubs; because those agents with higher scores do not learn from others (since there are no higher scoring others to learn from), they do not change their strategies, and their boldness and vengefulness remains close to the average, as shown in Figures 3(b) and 3(d). These results demonstrate that Mahmoud et al.'s algorithm is not effective in scale-free networks. Importantly, as the burden of punishment falls largely on hubs rather than outliers, hubs perform worst in the population. To address this, we modify the learning technique so that it can cope with the nature of scale-free networks, as discussed next.



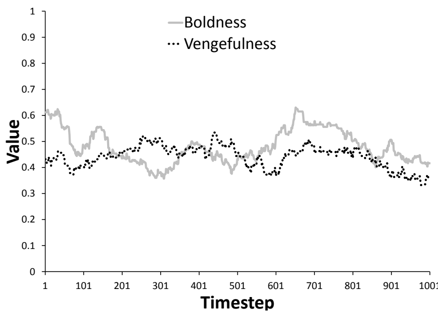
(a) Hubs - 100 timesteps



(b) Outliers - 100 timesteps



(c) Hubs - 1000 timesteps



(d) Outliers - 1000 timesteps

Fig. 3. Sample Run

3.1 Universal Learning

The algorithm proposed by Mahmoud et al. suffers from the limitation that it requires knowledge of the average score in the population in order for an agent to determine whether to modify its policies. However, since the aim of that work is to eliminate the unreasonable assumption of *omniscience*, by which agents are able to observe the private strategies of others, as well as observing all norm violations and punishments, it makes little sense to assume that agents have access to an average population score against which to compare themselves before deciding whether to modify their policies. For this reason, we consider an alternative approach, in which agents always modify their policies to improve performance, regardless of the behaviour of others, and only in relation to their own score. This modification is simple, and involves removing line 3 of Algorithm 3 (we do not show the new algorithm due to the simplicity of the change and space constraints).

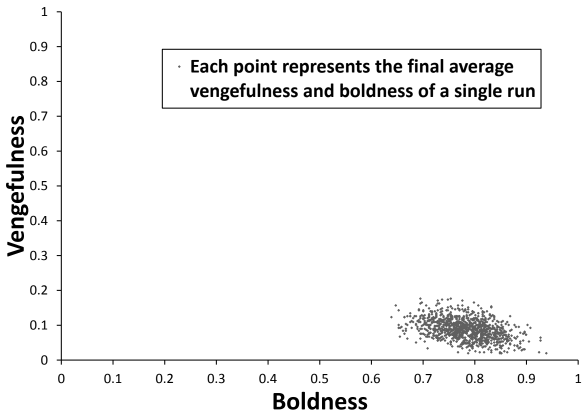


Fig. 4. Universal learning- Overall Result - 1000 Runs, 1000 Agents, 1000 Timesteps

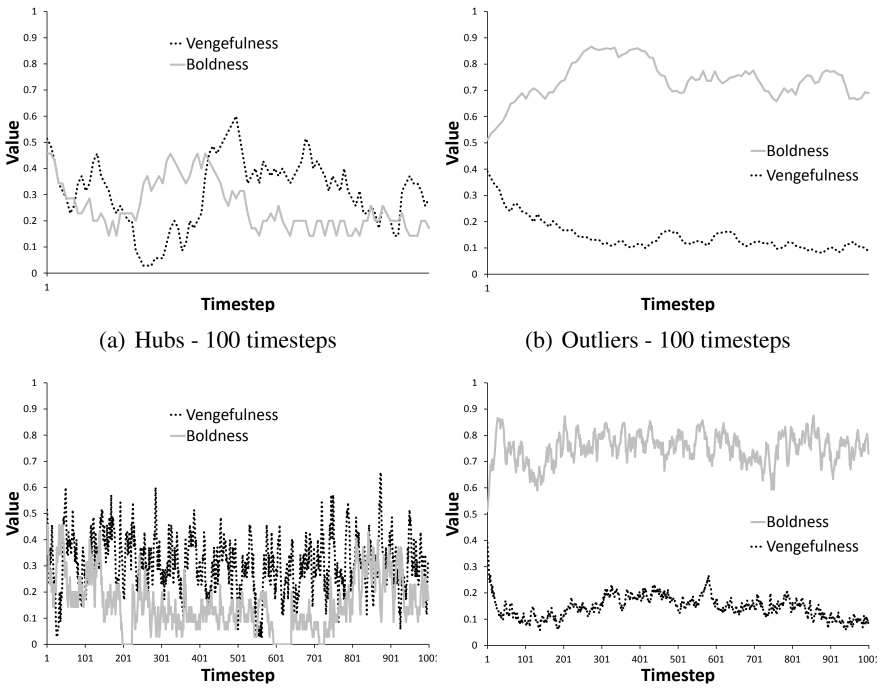
Experiments with this new approach give the results shown in Figure 3.1. Surprisingly, the results indicate norm collapse, as all runs end with high boldness and low vengefulness. By analysing the performance of the different types of agents, we are able to explain this behaviour; we illustrate by reference to a sample run for a hub in Figures 5(a) and 5(c), and a sample run for an outlier agent shown in Figures 5(b) and 5(d).

Outliers have few connections, but are connected to one or more hubs. When agents punish others, they pay an enforcement cost but risk metapunishment when they do not. However, since these outliers have very low connectivity, the risk of metapunishment is also very low, so they avoid punishing others and vengefulness consequently decreases. Metanorms are thus not effective here because of the lack of connectivity between agents. Outliers thus always have high boldness and low vengefulness levels. In addition, as we will see, the vengefulness of hubs also drops and is never higher than average, so agents can defect and gain benefit, without being punished by hubs. Outliers thus increase their boldness, causing norm collapse in the whole population.

In contrast to outliers, hubs are highly connected and apply punishments to many others, incurring high enforcement costs. To address this, they decrease their vengefulness, resulting in metapunishment from the many nodes to which they are connected, in turn causing hubs to increase vengefulness (but only to a mid-range level). In addition, because of the high boldness of outliers, there is a high rate of defection in the population, causing oscillation between mid-range and low vengefulness for the duration of the run. Boldness of hubs is kept low, however, due to the amount of punishment that the hubs are exposed to. Values for vengefulness and boldness are shown Figure 5(c).

3.2 Connection-Based Observation

Axelrod's original model considers a probability of being seen, and in the context of a fully connected network, this may be a reasonable basis on which to base a model.



(a) Hubs - 100 timesteps

(b) Outliers - 100 timesteps

(c) Hubs - 1000 timesteps

(d) Outliers - 1000 timesteps

Fig. 5. Universal learning - Sample Run

However, in the kinds of topologies we are concerned with, such as those that reflect the situations in peer-to-peer (P2P) networks or wireless sensor networks, for example, observation of the behaviour of others arises from the direct connection between agents. Thus, if a peer A is connected to another peer B, then A may be able to observe all communication from B. As a result, if B defects by, for example, not sharing files in the case of a file-sharing P2P network, this can be observed by A. To reflect this property in our model, Axelrod's probability of being seen requires replacing with the notion that each agent observes all actions of its direct neighbours. This modification to the model gives rise to rather different results. In particular, the results of running the model on a scale-free network, in Figure 3.2, show that all runs end in low boldness and low vengefulness, indicating that defection is very rare in the population because of the low boldness. In addition, punishment is not common since agents rarely punish defectors, due to their low vengefulness. To understand this better, the results of a 1000 timestep run, for outliers and hubs, are shown in Figures 7(d) and 7(c), respectively.

More specifically, Figures 7(b) and 7(d) show that outliers start the run by decreasing both vengefulness and boldness to a low level where they remain, with some small degree of fluctuation. Figures 7(a) and 7(c) suggest that hubs start the run by increasing their vengefulness to a high level and decreasing their boldness to a very low level. After a few timesteps, vengefulness decreases to a mid-range level, from which it decreases

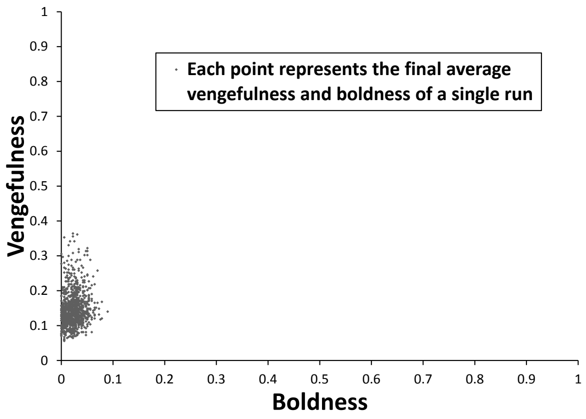


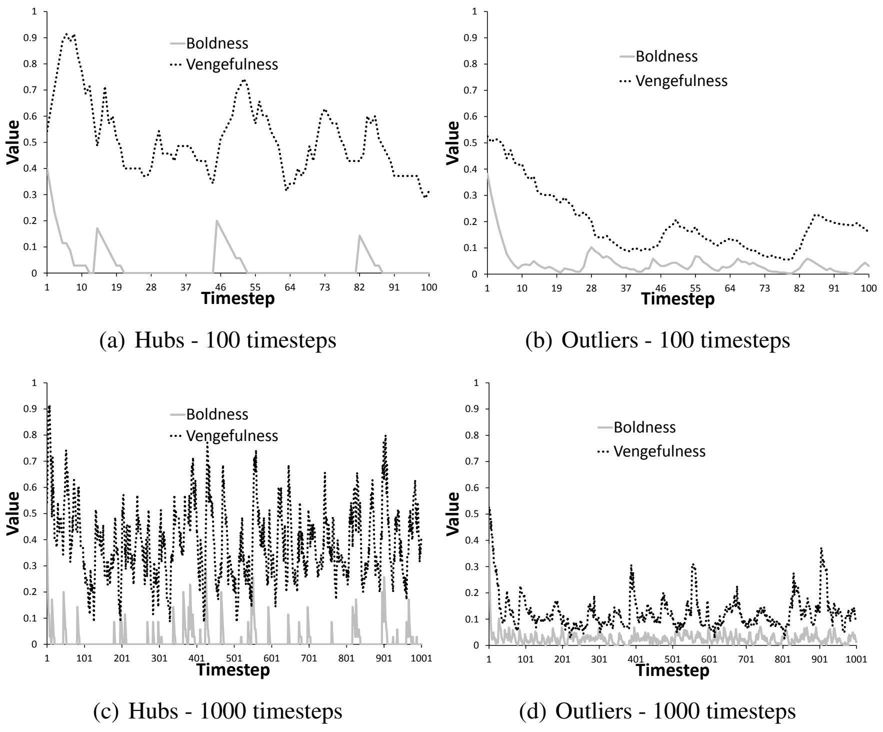
Fig. 6. Connection-Based Observation - Overall Result - 1000 runs, 1000 agents, 1000 timesteps

further to a low level. However, it does not stabilise there, since it moves up again, and this pattern is repeated throughout the run. Similarly, boldness initially decreases to zero and then jumps to a low level, before decreasing back to zero. Hubs thus have a fluctuating mid-range level of vengefulness, and a very low level of boldness.

There are two distinctive features that can be observed here, in contrast to the results obtained by the universal learning approach. First, hubs reach a high level of vengefulness, which is limited to mid-range vengefulness in the previous approach. This is mainly because the new technique raises the action observation probability to 100%, which allows a high possibility for metapunishment to occur and, as a result, forces hubs to increase their vengefulness to a high level. However, as before, this does not persist because of the high enforcement cost observed with such a high level of vengefulness. Second, the boldness of outliers is low here, mainly due to the combination of the high vengefulness among hubs and the 100% defection observation, which together produce sufficient punishments to force outliers to decrease their boldness.

4 Dynamic Policy Adaptation

As we have seen, universal learning has a negative impact on results, causing boldness to increase and vengefulness to decrease. However, a more important weakness is that the learning rate is uniform in the face of differing punishment levels: all agents use the same learning rate, regardless of how much utility gain or loss they suffer. Thus, for example, an agent that incurs a punishment score of -10 must modify its vengefulness to exactly the same degree as another agent whose punishment score is -999. While the direction of change is appropriate, the degree of change does not reflect the severity of the sanction; a more appropriate approach would change policy in line with performance. In this view, a very badly performing agent should modify its policy much more significantly than one that performs better. Dynamic policy adaptation can address this, bringing about changes to vengefulness and boldness that reflect performance. The key idea here is to measure the *level* of performance rather than just the *direction*, through



(a) Hubs - 100 timesteps (b) Outliers - 100 timesteps

(c) Hubs - 1000 timesteps (d) Outliers - 1000 timesteps

Fig. 7. Connection-Based Observation - Sample Run

comparison of an agent's actual utility, or *score* in our terms, and the maximum or minimum that could be obtained. We apply this to boldness and vengefulness in turn, but first introduce some notation. Let NDD be the number of available defection decisions, where each agent has multiple chances to defect in a single round (as specified earlier), NB_i be the number of i's neighbours, T be the utility gained from a single defection, and PC be the punishment cost representing the utility lost from being punished.

4.1 Boldness

In terms of boldness, the relevant part of the total score is the *defection score*, which can be either positive or negative, requiring consideration of both maximum and minimum possible values. The maximum possible defection score $MaxDS_i$ arises when an agent i always defects but is never punished, and the minimum defection score arises when the agent always defects and is always punished by all of its neighbours, as follows.

$$MaxDS_i = NDD \times T \tag{1}$$

$$MinDS_i = NDD \times (T + (NB_i \times PC)) \tag{2}$$

Then, in order to determine the degree of change to an agent's boldness, we must consider three different situations. First, when the defection score is positive (so that boldness should increase), the degree of change is determined by dividing the obtained defection score by the maximum possible defection score. Second, when it is negative, (so that boldness should decrease), the obtained defection score is divided by the minimum possible defection score. Finally, if the defection score is zero, no change is required.

$$FactorB_i = \begin{cases} \frac{DS_i}{MaxdDS_i} & \text{if } DS_i > 0 \\ \frac{DS_i}{MinDS_i} & \text{if } DS_i < 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Given this, we now need to determine how $FactorB$ can be used to change an agent's policy. In order to avoid dramatic policy movements that could lead to violent fluctuations, we limit the change that can be applied to a maximum value. In this case, the maximum is the difference between two levels as in Axelrod's original model, of $\frac{1}{7}$. Thus, an agent modifies its boldness in line with its DS, as follows, so that it can maximally change its boldness by one level (or by $\frac{1}{7}$) when $FactorB$ is 1.

$$B_i = B_i + \begin{cases} \frac{1}{7} \times FactorB_i & \text{if } DS_i > 0 \\ -\frac{1}{7} \times FactorB_i & \text{if } DS_i < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

4.2 Vengefulness

An agent modifies its vengefulness depending on whether it is valuable to punish others, determined by comparing the utility lost from punishing others (the punishment score, PS) against and the utility lost from not punishing them (the punishment omission score POS). If PS is greater than POS, agents increase vengefulness and decrease it otherwise. Clearly, the magnitude of this difference between these two values gives an indication of the degree of change that should be applied to vengefulness. For example, if PS is -24 and POS is -20, then the degree of decrease to V should be significantly lower than when PS is -600 and POS is -20. We call this difference $DiffV$:

$$DiffV_i = |PS_i - POS_i| \tag{5}$$

Since $DiffV$ is 1 or more (when the values are not equal), it cannot be used directly to update an agent's V value, because V must always lie between 0 and 1. It must thus be *normalised* so that it can be applied to V, for which we use a scaled value, $FactorV$; this is determined by dividing $DiffV$ by the minimum of PS and POS. Since both PS and POS are negative, their absolute value is used to obtain a positive value:

$$FactorV_i = \frac{DiffV}{|\min\{PS_i, POS_i\}|} \tag{6}$$

While this always produces a value between 0 and 1, it does not provide the same value for the same magnitude of difference. For example, if PS is -14 and POS is -20, we want $FactorV_i$ to be the same as when PS is -6 and POS is 0. We can achieve this by replacing the maximum of PS and POS with the maximum possible difference between PS and POS. This maximum difference is the difference from 0 (when there is no cost at all from punishing or from not punishing) to the greatest possible magnitude of PS or POS. In what follows, HPS represents the highest punishment score (the maximum in magnitude, and lowest in numerical terms — we use HPS to indicate the *highest* score to avoid ambiguity of minimum and maximum) that can be received by an agent punishing all of its neighbours for defection, and metapunishing all of its neighbours for not punishing all of their neighbours for defection.

To determine the value of HPS we need to consider both the punishment enforcement cost and the metapunishment enforcement cost. First, the highest (maximum in magnitude, but minimum numerically) *punishment* enforcement cost ($HPEC$) arises when all of an agent's neighbours defect and the agent punishes all of them:

$$HPEC_i = NDD \times NB_i \times EC \tag{7}$$

where EC is the enforcement cost of a single punishment. Similarly, the highest *metapunishment* enforcement cost ($HMPEC$) arises when all of an agent's neighbours do not punish all of their neighbours for defecting, and the agent metapunishes all of them:

$$HMPEC_i = NDD \times NBB_i \times EC \tag{8}$$

where NBB_i is the total number of neighbours of all of agent i's neighbours. HPS is thus defined as the sum of these two scores:

$$HPS_i = HPEC_i + HMPEC_i \tag{9}$$

In the same way, $HPOS$ is the highest (greatest in magnitude, lowest numerically) score that can be obtained when an agent does not punish any defectors, but is metapunished by all of its neighbours.

$$HPOS_i = NDD \times NB_i \times (NB_i - 1) \times PC \tag{10}$$

where the maximum number of defectors is all of an agent's neighbours (NB), the maximum number of metapunishers is the same but excluding the defecting agent, and PC is the punishment cost obtained from being metapunished (which is the same as for simply being punished). Given this, $FactorV$ can be calculated by dividing $DiffV$ by one of these values, as follows. If punishing brings a greater utility reduction than not punishing ($PS < POS$), then we use the highest punishment score HPS. Conversely, if $PS > POS$, then we use the highest punishment omission score $HPOS$. If there is no difference, then there is no change and $FactorV$ is equal to 0.

$$factorV_i = \begin{cases} \frac{DiffV_i}{HPS_i} & \text{if } POS_i > PS_i \\ \frac{DiffV_i}{HPOS_i} & \text{if } POS_i < PS_i \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

This guarantees that the change made to V is always the same given the same difference in scores, since both HPS and $HPOS$ are fixed for each agent. Moreover, this approach allows hubs to change much less quickly than outliers, because the highest (maximum in magnitude) scores for hubs are much higher than for outliers, so that the results achieved by using $FactorV$, and dividing by the difference in scores obtained for hubs, is much less than for outliers. As the learning algorithm suggests, an agent increases vengefulness when it finds that not punishing is worse than punishing, and it decreases vengefulness when the converse is true.

$$V_i = V_i + \begin{cases} \frac{1}{7} \times FactorV_i & \text{if } |PS_i| < |POS_i| \\ -\frac{1}{7} \times FactorV_i & \text{if } |PS_i| > |POS_i| \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

4.3 Example

To illustrate, assume that a hub A is connected to 20 other agents, and that an outlier B is connected to only 2 other agents (one being a hub). Like Axelrod's seminal experiments and without loss of generality, let $NDD = 4$ for all agents, since every agent has 4 chances to defect in each round. $EC = -2$ and is the same for all agents. Similarly, $PC = -9$ and again is the same for all agents. The temptation value for all agents, received when they defect, is $T = 3$. Finally, suppose that A's neighbours have 50 other distinct neighbours in total (summed over all neighbours), while B's neighbours have 20 other distinct neighbours (again, over all). This is summarised in Table 1. Given these values, we can determine the relevant values needed as follows. Starting with defection scores and from Equations 1 and 2 respectively, we obtain the following:

$$\begin{aligned} MaxDS_A = MaxDS_B &= 4 \times 3 = 12 \\ MinDS_A &= 4 \times (3 + (20 \times -9)) = -708 \\ MinDS_B &= 4 \times (3 + (2 \times -9)) = -60 \end{aligned}$$

In terms of punishment values, from Equations 7, 8 and 9, we obtain the following:

$$\begin{aligned} HPEC_A &= 4 \times 20 \times -2 = -160 \\ HMPEC_A &= 4 \times 50 \times -2 = -400 \\ HPS_A &= -160 - 400 = -560 \\ HPEC_B &= 4 \times 2 \times -2 = -16 \\ HMPEC_B &= 4 \times 20 \times -2 = -160 \\ HPS_B &= -16 - 160 = -176 \end{aligned}$$

Punishment omission scores using Equation 10 are as follows:

$$\begin{aligned} HPOS_A &= 4 \times 20 \times 19 \times -9 = -13680 \\ HPOS_B &= 4 \times 2 \times 1 \times -9 = -72 \end{aligned}$$

Using this information (Table 1), we can determine the decisions for specific situations. For example, at the start of each run, the population has average mid-range boldness and vengefulness (because of the uniform distribution function to generate initial policies). Now, suppose that both A and B also have mid-range boldness and vengefulness. If, after one round, both A and B defected twice (out of their four opportunities to defect),

Table 1. Example values for Agents A and B

Agent	Pos	NB	NBB	MinDS	MaxDS	LevB	HPS	HPOS	LevV
A	Hub	20	50	-708	12	1/7	-560	-13680	1/7
B	outlier	2	20	-60	12	1/7	-176	-72	1/7

they each gain twice the temptation value T. However, since A is a hub, suppose it is punished 22 times, much more than B, which is only punished twice. This is because the defection score of a hub with mid-range boldness is typically much worse than that of a similar outlier, mainly due to the difference in their number of neighbours, and the midrange vengefulness in the population. Thus, A has a defection score of 2×3 from defecting, plus $22 \times -9 = -198$ from being punished, giving $DS_A = -192$. Similarly, $DS_B = ((2 \times 3) + (2 \times -9)) = -12$.

Given these defection scores, the degree of change that each agent applies to its boldness can be calculated as follows. First, from Equation 3, $FactorB_A = \frac{-192}{-708} = 0.3$ and $FactorB_B = \frac{-12}{-60} = 0.2$. Now, using Equation 4, and since both DS_A and DS_B are negative, B_A is decreased by $0.3 \times \frac{1}{7} = 0.04$, and B_B by $0.2 \times \frac{1}{7} = 0.03$.

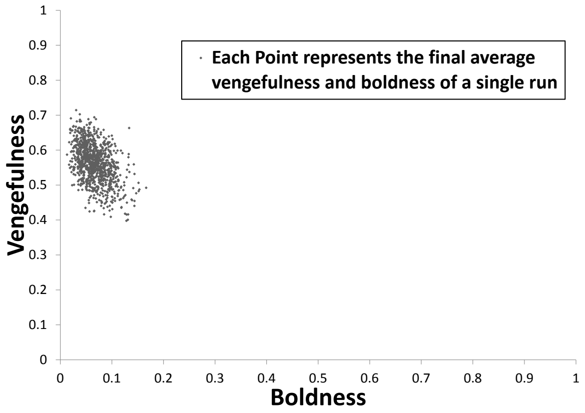
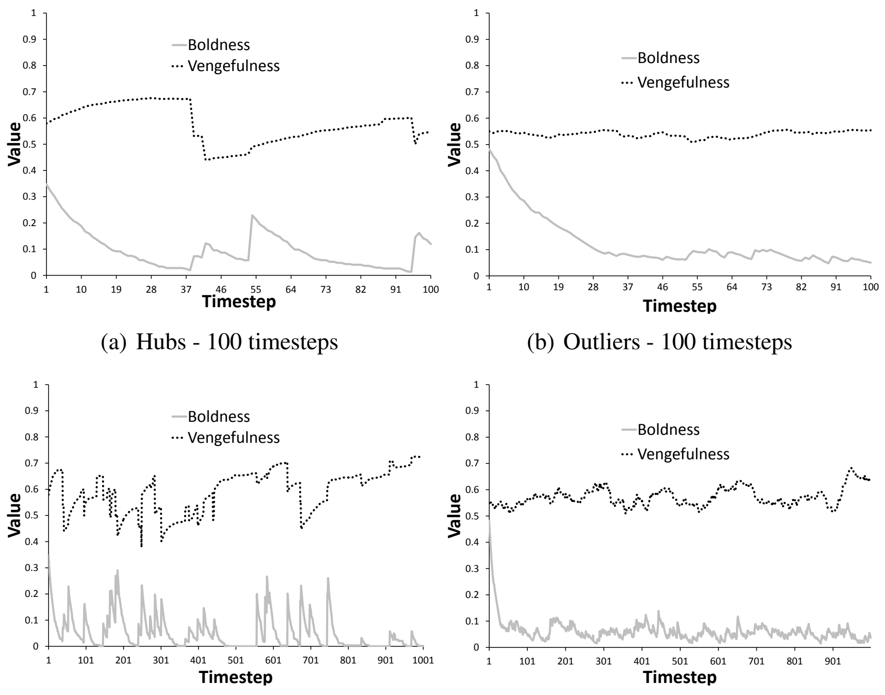


Fig. 8. Dynamic Policy Adaptation - Overall Result - 1000 Runs, 1000 Agents, 1000 Timesteps

In addition, if A punishes 35 other agents and metapunishes 16 more, and B punishes 10 other agents and metapunishes 4 more, their punishment scores are determined by multiplying the number of punishments issued by the enforcement cost EC: $PS_A = ((35+16) \times -2) = -102)$ and $PS_B = ((10+4) \times -2) = -28)$. Then, if A has spared 27 defectors and has been metapunished 6 times for each instance of omitting punishment, and B has spared only one defector and been metapunished just once, the punishment omission scores are calculated by multiplying the number of metapunishments by the punishment cost PC, as follows: $POS_A = (27 \times 6 \times -9) = -1458$ and $POS_B = (1 \times 1 \times -9) = -9$. Thus, by Equation 11, $FactorV_A = \frac{|-102-(-1458)|}{13680} =$



(a) Hubs - 100 timesteps (b) Outliers - 100 timesteps

(c) Hubs - 1000 timesteps (d) Outliers - 1000 timesteps

Fig. 9. Dynamic Policy Adaptation - Sample Run

0.1 and $FactorV_B = \frac{|-28-(-9)|}{96} = 0.2$. Then, since $PS_A > POS_A$, A increases its vengefulness V_A by $0.1 \times \frac{1}{7} = 0.014$ according to Equation 12). Similarly, since $PS_B < POS_B$, B decreases its vengefulness by $0.2 \times \frac{1}{7} = 0.03$.

4.4 Experimental Results

To determine the effect of introducing dynamic policy adaptation, we ran experiments, similar to the previous experiments, on the new model, and giving the results shown in Figure 4.3. As can be seen in the figure, all runs result in populations with low average boldness and moderate vengefulness. As before, more detail on the evolution of average boldness and vengefulness for hubs and outliers was provided by examining runs of individual agents, as shown in Figure 9 , which confirm that outliers converge to a state of low boldness and moderate vengefulness consistently, while hubs do so with intermittent deviations. As before, hubs increase vengefulness and decrease boldness, though much more slowly now. However, at regular intervals, there are sudden increases to boldness, accompanied by a change in vengefulness, as a result of the exploration of the algorithm. This phenomenon occurs in all models in this paper, and is visible here due to the limited number of timesteps, but has no impact on the results of the dynamic policy adaptation.

5 Conclusion

Norm emergence is an important and valuable phenomenon that has applications to self-organising systems such as peer-to-peer networks or wireless sensor networks in which there is no interference from any central or outside authority. While there has been much work on this phenomenon (as discussed earlier), and even some on its application to different topological structures, there has been inadequate consideration of how to establish norms in scale-free networks. Indeed, some mechanisms have been shown not to succeed in these topologies. In response, this paper provides an effective means of overcoming the problems arising from asymmetric connections of hubs and outliers.

In particular, our results show that in scale-free networks, Axelrod's basic metanorm model is not effective, nor is Mahmoud et al.'s attempt to overcome this for other topologies. Our simulations suggest that poorly connected agents receive little discouragement from defecting while hubs are discouraged from enforcing norms through high enforcement costs. In response, we have modified the experimental setting to be more consistent with the nature of distributed systems of partially connected nodes, bringing an even more serious breakdown in norm emergence, but also subsequently addressed this through a dynamic policy adaptation mechanism. In this way, agents are able to change their policy in proportion to the punishments they receive, allowing them to adapt proportionally, and to maintain the policy values that sustain norm establishment.

In terms of future work, we plan to conduct a more detailed analysis of the effect of different levels of the probability of observation on the results of the model. In addition, we aim to develop an efficient adaptive punishment approach that allows punishment to be applied according to the specific case at hand, so that agents with different degrees of defection will be punished accordingly. This is important so that the punishment is no greater than needed, and will not overly constrain behaviour (limiting the functionality of the underlying system, which is undesirable). We believe that the dynamic policy adaptation technique that has been introduced in this paper provides a solid grounding for just such an adaptive punishment approach.

References

1. Adar, E., Huberman, B.A.: Free riding on gnutella. First Monday 5(10) (2000)
2. Axelrod, R.: An evolutionary approach to norms. The American Political Science Review 80(4), 1095–1111 (1986)
3. Barabasi, A.L., Albert, R.: Emergence of Scaling in Random Networks. Science 286(5439), 509–512 (1999)
4. de Pinninck, A.P., Sierra, C., Schorlemmer, W.M.: Friends no more: norm enforcement in multiagent systems. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 640–642 (2007)
5. Delgado, J., Pujol, J.M., Sangüesa, R.: Emergence of coordination in scale-free networks. Web Intellgence. and Agent Systems 1, 131–138 (2003)
6. Epstein, J.M.: Learning to be thoughtless: Social norms and individual computation. Computational Economics 18(1), 9–24 (2001)
7. Flentge, F., Polani, D., Uthmann, T.: Modelling the emergence of possession norms using memes. Journal of Artificial Societies and Social Simulation 4(4) (2001)

8. Galán, J.M., Latek, M.M., Rizi, S.M.M.: Axelrod's metanorm games on networks. PLoS ONE 6(5), e20474 (2011)
9. Mahmoud, S., Keppens, J., Luck, M., Griffiths, N.: Norm establishment via metanorms in network topologies. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 3, pp. 25–28 (2011)
10. Mahmoud, S., Keppens, J., Luck, M., Griffiths, N.: Overcoming omniscience in axelrod's model. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 3, pp. 29–32 (2011)
11. Savarimuthu, B.T.R., Cranefield, S., Purvis, M.K., Purvis, M.: Norm emergence in agent societies formed by dynamically changing networks. In: Proc. 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 464–470 (2007)
12. Savarimuthu, B.T.R., Purvis, M., Purvis, M., Cranefield, S.: Social Norm Emergence in Virtual Agent Societies. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2008. LNCS (LNAI), vol. 5397, pp. 18–28. Springer, Heidelberg (2009)
13. Sen, O., Sen, S.: Effects of Social Network Topology and Options on Norm Emergence. In: Padget, J., Artikis, A., Vasconcelos, W., Stathis, K., da Silva, V.T., Matson, E., Polleres, A. (eds.) COIN 2009. LNCS (LNAI), vol. 6069, pp. 211–222. Springer, Heidelberg (2010)
14. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence 73(1-2), 231–252 (1995)
15. Villatoro, D., Sen, S., Sabater-Mir, J.: Topology and memory effect on convention emergence. In: Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technologies, pp. 233–240.
16. Walker, A., Wooldridge, M.: Understanding the Emergence of Conventions in Multi-Agent Systems. In: Lesser, V. (ed.) Proceedings of the First International Conference on Multi-Agent Systems, pp. 384–389. MIT Press (1995)

Norm Contextualization

Jie Jiang, Huib Aldewereld, Virginia Dignum, and Yao-Hua Tan

TPM, TU Delft, The Netherlands
{jie.jiang,h.m.aldewereld,m.v.dignum,y.tan}@tudelft.nl

Abstract. Agents interact with each other regulating by a set of norms which is expressed at different levels of abstraction that capture different contexts and operationalizations. Current normative frameworks deal with norm operationalization, yet few consider the contextual aspects of norms. Moreover, most frameworks are based on the independent evaluations of norms, which makes it difficult to evaluate interrelated effects of different norms and contexts. In this paper, we propose *Norm Nets* as a formalism to capture the structure of contextualized norm sets. This formalism will enable (1) the analysis of interrelations between norms, (2) the contextualization of normative statements, and (3) the verification of properties of interrelated norms. We apply this framework to a case study taken from the domain of international trade.

1 Introduction

Open regulated Multi-Agent Systems (MAS) assume that agents are subjected to norms (explicit or implicit) that regulate their behavior. A large number of research in normative agent systems focuses on how agents can decide whether to comply with norms (e.g., [12,6,8]). Another research area concerns consistency aspects of the normative structure in MAS (e.g., [9,19,14]). However, the traditional way of organizing norms does not emphasize their interrelations and application environments at a large scale, which is very important in nowadays' business operations. For example, in the domain of international trade, different regulations would be applied when an importer imports the same kind of goods from different countries. The changes will result in cost if the importer could not follow the right set of regulations with respect to its associated business environment. Moreover, given a specific situation, the set of applicable regulations are not independent with each other but have different interrelationships. A typical example of such interrelationships is a regulation and its sanction, indicating a conditional and exclusive relation between two regulations.

In order to explore the interdependencies between norms and their application environments, this paper proposes an approach to represent and analyze sets of norms that takes into consideration both the interrelationships between different norms and the context of their application. More importantly, the explicit representation of institutional contexts on norms facilitates a contextual refinement normative structure, which supports norm design at multiple levels of abstraction. Our approach is different from those based on deontic reasoning, as we do not aim at identifying the deontic consequences of actions. Instead, we try to detail how norms can be abstracted from reality and organized in a structured way to facilitate contextualization and compliance checking.

To illustrate our proposal, we adopt a scenario in the domain of international trade in which a trading network may include a variety of entities (e.g., software, organizations and people) that are largely autonomous, geographically distributed, and heterogeneous in terms of their operating environment, culture, social capital, and goals. In this domain, agents represent real interests and real entities, i.e., different agents have different owners, goals, interests, and preconditions for collaboration. For example, importers are motivated by profit and quality of products, while customs authorities are motivated by safety and security concerns. At any given moment, most agents will be conditioned by different regulations and norms, originating from different institutional contexts.

In general, our contributions are three-fold. First, we use *context* to organize norms in such a way that the interrelationships between norms and their application environments are structured at multiple abstraction levels. Second, we formalize a methodology of practicing contextualization, with which designers can easily build up their own context-aware normative models. Third, in order to ground the operational aspects of our framework, we design a mapping method which translates models of our framework to Colored Petri Nets (CPNs) that can be used for checking consistency of the designed models.

The rest of this paper is organized as follows. In the next section, we present a simplified scenario from the domain of international trade. In section 3, we introduce a normative model *Norm Nets*, which is further extended to include contexts refinement in section 4. Section 5 present the mapping from Norm Nets into CPNs. Related work is discussed in Section 6. Finally, we conclude our work and raise directions for future work in section 7.

2 Scenario

In this section, we introduce a simplified scenario in the field of international trade concerning the issues of *origin of goods* in importing and exporting under EU's regulation [1]. This scenario is used throughout the paper to explain our proposal.

Origin is the "economic" nationality of goods in international trade, which is generally required to be indicated in the export/import documents and governmental submissions when transporting goods from one country to another. This is checked in different ways. For example, certificates have to be presented when importing goods from a certain country. Such certificates usually contain the country of origin, the national goods code, etc. These should match the information that is listed on the invoice and the packing list. According to the characteristics of the trading goods, there are two types of origin: *non-preferential origin* and *preferential origin.* Regulations for non-preferential origin are used for all kinds of commercial policy measures such as anti-dumping duties, countervailing duties, trade embargoes, safeguard and retaliation measures, quantitative restrictions, but also for some tariff quotas, trade statistics, public tenders, and origin marking etc. In addition, the EU's export refunds in the framework of the Common Agricultural Policy are often based on non-preferential origin. Preferential origin is conferred on goods from particular countries, which have fulfilled given criteria. In order to obtain preferential origin, it is generally required that the goods be wholly obtained or have undergone specific processing activities. Preferential origin confers certain tariff benefits (entry at a reduced or zero rate of duty) on goods traded between countries which have agreed such an arrangement or where one side has granted it autonomously.

3 Normative Structure

Usually, norms are perceived as a set of dos and don'ts. For example, in sociology, a norm is considered as a rule or standard of behavior accepted by members within a society or group[1]. In economics, a norm is a model of what should exist or be followed, or an average of what currently does exist in some context, such as an average salary among the members of a large group [7]. Formalized by E. Ostrom [16], a norm can be defined by the ADICO syntax which describes who (Attribute) is obliged/forbidden/permitted (Deontic) to do or achieve what (aIm), when and where (Condition), otherwise (Or else) leading to consequences of violation.

To reflect the reality, many MAS variants tried to incorporate norms as a formal specification of deontic statements that aim at regulating the behavior of software agents and the interactions among them [2,11,12], which focus more on the operational level. While, the ADICO syntax is a natural expression of norms which (1) provides a clear description of opportunities and constraints that create expectations about actors' behavior, and (2) makes it intuitive for the actors to understand their dos and don'ts. Therefore, we follow the ADICO syntax with adaptation to a MAS-integrable definition of norm, as shown in Definition 1.

Definition 1 (norm). *A norm is defined as a tuple* $n = (role, deontic, action, condition)$ *such that:*

- *role indicates the organizational position to whom the norm applies;*
- *deontic is one of the three modal verbs "may" (*$\underline{P}$*ermitted), "must/should" (*$\underline{O}$*bliged) and "must not/should not" (*$\underline{F}$*orbidden);*
- *action specifies the particular action to which the deontic is assigned;*
- *condition describes when and where the norm holds.*

If a norm does not specify a particular role or condition, the default value is for all participants, or at all times and in all places covered by that norm. The building blocks of our norm definition are expressed using lowercase (with or without subscripts). Corresponding expressions using uppercase indicate sets of a particular element, e.g., $ROLE$ indicates a set of roles such that $role \in ROLE$.

[1] cf. Encyclopedia Britannica: `http://www.britannica.com/`

When a set of norms are imposed on a multi-agent system, they are usually not independent of each other but interrelated in different ways. For example, as an undergraduate student in the Netherlands, one should on the one hand pay the tuition fee, and on the other hand, obtain a certain amount of academic credits. This indicates that both norms have to be complied with when enrolled as a Dutch undergraduate student. Another typical relation between two norms is a norm and its sanction. In particular, obligations and prohibitions may have corresponding sanctions when the norms are violated. Sanctions are norms which will be triggered when violations are detected, indicating a conditional and exclusive relation between norms. To model the possible relationships between norms, we introduce three logical operators *AND*, *OR*, and *OE* (representing Or else). Taken from the scenario introduced in section 2, the following are examples of how norms are interrelated.

- *AND*(n_1, n_2): both norms should be met.
  (ex1) (n_1, n_2) [*role*: The customs authorities] [*deontic*: should] (n_1) [*action*: grant to the approved exporter a customs authorization number]. *AND* (n_2) [*action*: monitor the use of the authorization by the approved exporter].
- *OR* (n_1, n_2): choice between the two norms.
  (ex2) (n_1), (n_2) [*role*: The customs authorities] [*deontic*: may] [*action*: exceptionally issue a certificate of origin] (n_1) [*condition*: after exportation of the products to which it relates, if the certificate of origin was not issued at the time of exportation because of errors]; *OR* (n_2) [*condition*: after exportation of the products to which it relates, if it is demonstrated to the satisfaction of the competent governmental authorities that a certificate of origin was issued but was not accepted at importation for technical reasons].
- *OE* (n_1, n_2) indicates the two norms are exclusive and conditional (only when n_1 is violated can n_2 be met).
  (ex3) (n_1) [*role*: The approved exporter] [*deontic*: should] [*action*: offers all guarantees necessary to verify the originating status of the products], [*condition*: irrespective of the value of the products concerned]. *OE* (n_2) [*role*: The customs authorities] [*deontic*: should] [*action*: withdraw the authorization] [*condition*: at any time where the approved exporter no longer offers the guarantees necessary to verify the originating status of the products].

As seen from these examples, the interrelationships between norms are attached to different normative components. For instance, the *AND* relation of the two norms in (**ex1**) is attached to two different *actions* while refers to the same *role* and *deontic*, and the *OR* relation of the two norms in (**ex2**) is attached to two different *conditions* while refers to the same *role*, *deontic* and *action.* However, there is no restriction that an interrelationship between two norms is attached to a specific normative component.

Furthermore, the application of such a set of interrelated norms is usually associated with a certain institutional environment, representing which situations are constrained by the norms. For this purpose, we use an explicit representation of *contexts* to characterize the situations where a set of norms is applied. Zimmermann et al. [20] proposed a formal structure of context information, which constricts and clusters this information into five fundamental categories, i.e., individuality, time, location, activity and relations. The *individuality* category contains properties and attributes describing an entity. The *activity* category covers all tasks the entity may be involved in. The *location* and *time* categories provide the spatio-temporal coordinates of the respective entity. Finally, the *relations* category represents information about any possible relation an entity may establish with another entity.

This paper does not focus on how to model contexts but tries to explore an effective way of using contexts to organize norms. Therefore, we only give an abstract definition of *Context* based on the context structure proposed by Zimmermann et al.

Definition 2 (context). *A context c is a set of states defined by predicates concerning aspects such as individuality, time, location, activity, relations.*

For example, a context might be characterized by a certain location, indicating that the norms applied in this context are mainly concerned with specific spatio coordinates. Contexts won't change the specification of norms but provide additional information about the situations in which the norms are applied from a higher-level perspective.

Note that the *condition* of a single norm is only a local prerequisite of when and where the norm holds, e.g., (importers should submit importing declaration) [*condition*: before goods arrive at the EU boarder] is a specific description of the deadline for this norm, while a *context* characterizes the situations in which a set of interrelated norms are applied from a broader perspective, e.g., [*context*: preferential origin] characterizes the situation in which norms concerning beneficial treatments for certain countries are applied. That is, conditions are at the level of individual norm specifications and usually represent situational differences between different norms, while contexts are at the level of collective norm sets and indicate situational commonalties of a norm set.

Putting all these together, we formalize the definition of *Norm Net* that not only represents the interrelationships between norms but also reflects their application environments.

Definition 3 (Norm Net). *A Norm Net* $NN = (c, NS)$*, where*

- c *indicates the context of the norm net, and*
- $NS = n$*, or* $NS = AND(NS_i, NS_j)$*, or* $NS = OR(NS_i, NS_j)$*, or* $NS = OE(NS_i, NS_j)$ *where* n *is a norm,* NS_i*,* NS_j*, and* NS *are norm sets.*

Each norm net is associated with an institutional context which describes all the situational elements that characterize the application environment of a normative structure. Making the context explicit enables actors to control the evolution of the norm net, to accommodate compliance and resolution of conflicts from higher-level views. A norm set *NS* can be a single norm or a nested structure composed of norms, which are connected by the three operators AND, OR, and OE in a certain context. The norms and their sanctions are exclusive and conditional, i.e., one either conforms to the norms or accepts the sanctions when violating the norms. This is reflected by the semantics of OE operator. (ex3) shows an example of this situation where an obligation is connected with its sanction.

Figure 1 presents a graphical construction of a norm net $NN_1 = (c_1, NS_1)$, represented as an oval. NS_1, represented as a rectangle, is composed of two norm sets NS_2 and NS_3 connected by AND. Similarly, NS_2 is another AND connection of two sub norm sets NS_4 and NS_5 while NS_3 is an OR connection of two sub norm sets NS_6 and NS_7. Connected by an OE, NS_8 with NS_9 as the consequence of not following NS_8 build up NS_4. Specifically, we use dashed lines to indicate the consequence NS_9. As can be seen, the proposed framework enables a modular way of representing the interrelationships between norms in a specific context.

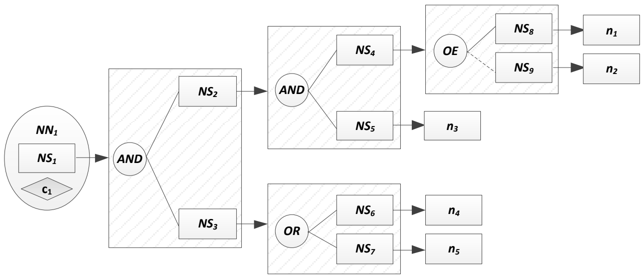


Fig. 1. An example of Norm Net

4 Contextualization

Laws and regulations are a system of textual rules and guidelines that are enforced through social institutions to govern behavior. They are specified as a normative structure, which describes the expectations and boundaries for agent behavior. We have already presented the representation of norms using *norm nets* in Definition 2 to capture the declarative meaning of the laws/regulations and also the relations between them. However, in real world domains, norms are not specified at a single level of abstraction. Usually, laws/regulations are first issued at a higher abstraction level stating the dos, don'ts and sanctions to regulate actors' behavior. Based on this abstract set of norms, elaboration will be conducted according to the specific characteristics and requirements of different situations, i.e. their application environments, which results into sets of contextual norms. This elaboration process facilitates detailed explanation of abstract norms in a concrete implementing environment.

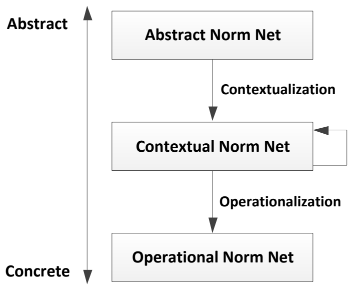


Fig. 2. Contextualization and operationalization

Our approach depicts three modeling layers of norms from abstract statements to concrete operations as shown in Figure 2. It starts from an abstract norm net which describes the expectations and boundaries for agent behavior in general. At this layer, norm specification is abstract and assumed to be stable throughout the life cycle of systems. The second layer uses contexts to organize the interrelated norms in different application environments derived from the abstract norm net. That is, the abstract norm net refers to a set of contextual norm nets which give more specific information on the roles, actions, conditions and the relations between the elaborated norms. Moreover, a contextual norm net can again refers to sets of contextual norm nets in a recursive manner, which enables a flexible normative structure and facilitates norm designing at different abstraction levels. In this way, the contexts of these norm nets establish a refinement relation captured in Definition 4.

Definition 4 (context refinement relation). *A context c' refines a context c, denoted as $c' \preceq c$ iff $c' \subseteq c$ assuming the ontologies of c' and c are unified.*

Note that in a context refinement relation $c' \preceq c$, the ontology used in c' may be more concrete than that in c (e.g., where in c one may talk about *vehicles* while in c' one may talk about *cars*). The unification of contexts is done via "counts-as" statements [3,7].

Given this context refinement relation, norm nets can be structured at multiple abstraction levels through contextualization defined in Definition 5.

Definition 5 (contextualization). *Given a context refinement relation $c' \preceq c$, a contextualization can be defined between two norm nets $NN' = (c', NS')$ and $NN = (c, NS)$ such that*

- *NS' elaborates NS with refined normative components, or*
- *NS' adds new norms to NS, or*
- *NS' removes some of the norms from NS, or*
- *NS' elaborates the interrelations between norms from NS.*

Definition 4 and 5 are reflections of the laws/regulations in practice. In this sense, contextual norm nets describe what properties should the concepts have from the specification of the abstract norm net to the refined contexts. For example, whether a document should be considered as a required certificate in international trade depends on the context in which the concept of *certificate* is used. A required certificate for importing fruit from China to the EU might not *counts as* a required certificate for importing textile.

In general, a norm net can have multiple contextualizations with respect to different contexts while different norm nets may be referred to in one contextualization. Moreover, there is no clear boundary between two contexts, i.e., the contexts of different norm nets may overlap, e.g., a context of the regulations for importing goods from Asia and another context of the regulations for importing textile products.

Finally, at the third layer, based on the contextual norm nets which contain enough information of the dos and don'ts in a specific situation, the norms will be extended with operational aspects to capture the operational meaning of the norms such as how the violation is detected (detection mechanism), and what can be done by the institution to repair the violation and minimize the negative influence[2]. Actors only need to reason about the norms at the most concrete level but the process of contextualization helps them to identify the applicable norms according to their runtime environment.

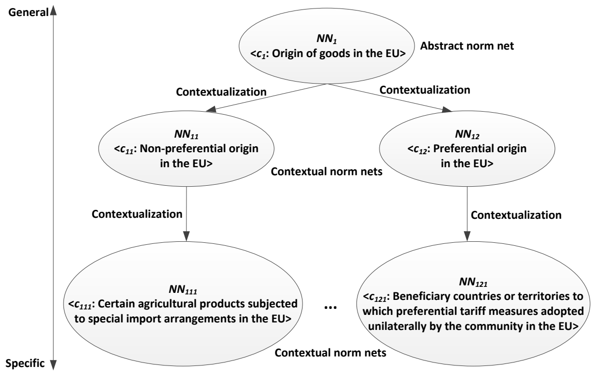

General
Specific
NN_1 <c_1: Origin of goods in the EU>
Abstract norm net
Contextualization
Contextualization
NN_{11} <c_{11}: Non-preferential origin in the EU>
NN_{12} <c_{12}: Preferential origin in the EU>
Contextual norm nets
Contextualization
Contextualization
NN_{111} <c_{111}: Certain agricultural products subjected to special import arrangements in the EU>
...
NN_{121} <c_{121}: Beneficiary countries or territories to which preferential tariff measures adopted unilaterally by the community in the EU>
Contextual norm nets

Fig. 3. Contextualization in the scenario

Continuing with the scenario, Figure 3 shows how the normative structure is built into a set of norm nets in a hierarchy from general to specific through contextualization. At the top level, a norm net NN_1 specifies a general set of regulations applied in the context c_1 "origin of goods in the EU". The norms at this level consist of coarser-grained components and provide an abstract view of the domain. As stated in Section 2, goods with different types of origin will be treated differently, which results in two exclusive sub contexts c_{11} "non-preferential origin in the EU" and c_{12} "preferential origin in the EU". For goods of preferential origin, more constraints as well as benefits are identified for exporters and importers in NN_{12}, which can be further contextualized. For example, a contextualization NN_{121} is built for certain beneficiary countries and territories to which preferential tariff measures are adopted. Similarly for goods of non-preferential origin, more specific regulations are formulated in NN_{11} and a further contextualization NN_{111} is constructed for certain agricultural products subjected to special import arrangements.

This norm refinement relation through contextualization is not only a natural reflection of how norms are evolved in real life but also makes it easier for actors to recognize their dos and don'ts according to their runtime environments. Moreover, norm nets are distributed into well-defined reusable components and enable hiding of details in a consistent way.

Based on the normative structure shown in Figure 3 and the practical regulations in the EU, we illustrate the contents of the norm nets abstracted from the scenario, which have been partially shown as $ex1, ex2, ex3$ in Section 3. However, due to space limitations, we can only present a small part for explanation.
$\mathbf{NN_1} = (\mathbf{c_1}, \mathbf{NS_1})$ where

- c_1 = "origin of goods in the EU",
- $NS_1 = AND(AND(n_1, n_2), OE(n_3, n_4))$, where
  - n_1: [*role*: Exporters] [*deontic*: should] [*action*: apply for certificate of origin] [*condition*: when exporting goods to the EU].
  - n_2: [*role*: The customs authorities] [*deontic*: should] [*action*: issue certificate of origin to the qualified applicants].
  - n_3: [*role*: Importers] [*deontic*: must] [*action*: present Customs with a specific origin documents] [*condition*: at the moment of import].
  - n_4: [*role*: The customs authorities] [*deontic*: should] [*action*: reject the import] [*condition*: when the origin documents cannot be presented].

$\mathbf{NN_{11}} = (\mathbf{c_{11}}, \mathbf{NS_{11}})$ where

- c_{11} = "non-preferential origin in the EU",
- $NS_{11} = OE(AND(AND(n_{a1}, n_{a2}), OR(n_{a3}, n_{a4})), n_{a5})$, where
  - n_{a1}: [*role*: The certificate of origin] [*deontic*: should] [*action*: measure 210×297 mm].
  - n_{a2}: [*role*: The certificate of origin] [*deontic*: should] [*action*: allow a tolerance of up to minus 5 mm or plus 8 mm in the length].
  - n_{a3}: [*role*: The certificate of origin] [*deontic*: should] [*action*: be printed in one or more of the official languages of the Community],
  - n_{a4}: [*role*: The certificate of origin] [*deontic*: should] [*action*: be printed in any other language] [*condition*: depending on the practice and requirements of trade].
  - n_{a5}: [*role*: The certificate of origin] [*deontic*: should not] [action: be approved] [condition: when it is not in the prescribed format].

$\mathbf{NN_{12}} = (\mathbf{c_{12}}, \mathbf{NS_{12}})$ where

- c_{12} = "preferential origin in the EU",
- $NS_{12} = AND(AND(n_{b1}, ex1), AND(ex2, ex3))$, where
  - n_{b1}: [*role*: The competent governmental authorities of the beneficiary country] [*deontic*: should] [*action*: ensure that certificates and applications are duly completed].

$\mathbf{NN_{111}} = (\mathbf{c_{111}}, \mathbf{NS_{111}})$ where

- c_{111} = "certain agricultural products subject to special import arrangements in the EU",
- $NS_{111} = OE(AND(AND(n_{a1}, n_{a2}), n_{a3}), n_{a5})$

From the description above, we can see that the norms in NN_1 only give a general idea about the regulations concerning the origin of goods in the EU. While in NN_{11} and NN_{12}, more specific norms are given in terms of descriptions about roles, actions and conditions such as the format of certificate of origin, the expected behavior of the approved exporter and under which conditions the customs should withdraw the authorization. Specifically, we can find the similarities and differences between the norms in NS_{11} and NS_{111}, which indicate that contextualization may not only add detailed information but also make changes.

5 Verification

To enable consistency and compliance checking of norm nets, we introduce a verification based on the mapping to the Colored Petri Nets.

5.1 Colored Petri Nets

CPN is a graphical language for modeling and validating distributed systems or systems in which concurrency plays a major role [13]. Not only do CPNs model the states of a system and the events that change the system from one state to another, but also replace tokens by data objects of programming languages.

Definition 6 (CPN). *A CPN is defined as a tuple ($\sum$, P, T, A, N, C, G, E, I) where: $\sum$ is a finite set of non-empty types, also called color sets; P is a finite set of places; T is a finite set of transitions; A is a finite set of arcs; $P \bigcap T = P \bigcap A = T \bigcap A = \Phi$; N is a node function defined from A into $P \times T \bigcup T \times P$; C is a color function defined from P into $\sum$; G is a guard function defined on T; E is an arc expression defined on A; I is an initialization function defined on P.*

In CPNs, places indicate states while transitions indicate actions. A place is a node where tokens from a specified color set may reside, and this color set is determined by the color function. Transitions are also represented as nodes. For each transition, a Boolean expression called a *guard* can be specified to restrict the conditions under which the transition can occur. Places may contain a natural number of tokens. A distribution of tokens over the places of a CPN is called a marking. The initial marking of a CPN specifies the initial load of tokens. An arc represents an input or output relationship between a place and a transition. The actual amount and the colors of tokens moved are specified by the corresponding arc expression. Based on the current marking, the guards can calculate which transitions are enabled with respect to which bindings. The bindings indicate the variables in the arc expressions. If there is no conflict with other transitions, an enabled transition may fire, whereby corresponding tokens are removed from the input places of the transition and added to the output places, as specified by the arc expressions. The transitions can be seen as patterns of behavior while the actual binding determines the details of the behavior. The number of tokens moved along an arc depends on the actual binding.

5.2 Mapping to CPNs

Norm Mapping. The mapping makes use of correspondences between the components in norms and the elements in CPNs.

$$ROLE \rightarrow \sum, \quad ACTION \rightarrow T, \quad CONDITION \rightarrow G$$

Roles in norms are mapped to the color sets in CPNs so that colored tokens correspond to role enacting agents in MAS. Actions in norms are mapped to the transitions in CPNs while conditions in norms are mapped to the guard functions in CPNs. Thus, only when the condition of a norm holds can the corresponding action specified in the norm be permitted, obliged or forbidden. Places in CPNs indicate the states of the role enacting agents, i.e., their behavior status in terms of norms. For the three deontic representations in norms, we use different building blocks with special places and transitions shown in Figure 4.

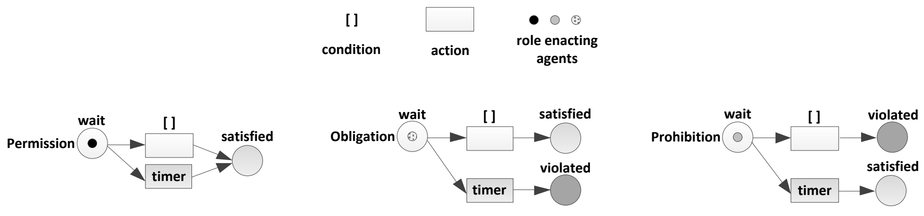


Fig. 4. Basic components in norms mapping to CPN

The CPN model of a norm starts with a place of *wait* which indicates the instantiation of the norm and has a color set indicating the role in the norm. Once the condition of the norm is satisfied, the agents in that place are able to perform the specified action. Permissions specify what might be done and won't lead to sanctions. Therefore, no matter the actions specified in a permission are performed or not, the final state of the permission will be *satisfied.* Obligations and prohibitions specify the actions that must and must not be done otherwise sanctions might be imposed, in the sense that the final state will either be *satisfied* or *violated.*

However, we cannot determine that someone does not follow a norm by simply saying that the action specified in the norm is not performed. In practice, whether a norm is satisfied or violated is normally determined within a certain period of time, e.g., the life cycle of an interaction scene. That is, the state of the norm is changed either because of the action specified in the norm is performed or the norm expires (a certain period passes). For this purpose, we adopt a special kind of transitions called *timer* that can be used to change the state of the norm when it expires and at the same time the action specified in the norm is not performed.

For permissions, both the action of the norm and the timer are connected to the place of *satisfied*, indicating that there are two ways to achieve this state: either the permitted action is performed by the agent or the timer is fired (i.e., the permitted action has not been performed when time is up).

Obligations indicate that agents should perform the specified actions and if this is the case, the token moves to the place of *satisfied.* But when the obliged action in the norm is not performed and the timer is fired, the token moves to the place of *violated.*

Prohibitions are a reverse logic of obligations. If the forbidden actions are performed, the corresponding tokens will move to the place of *violated.* But when the forbidden actions are not performed during the specified period of time, the corresponding tokens will move to the place of *satisfied.*

The description above only captures the mapping of individual norms. For norm nets, it requires a mechanism of representing different relationships between norms. To this end, we use extra elements for the mapping.

Norm Net Mapping. As an example, we model the norm net NN_1 of the scenario as shown in Figure 5. When the corresponding CPN is instantiated, all the tokens, i.e., all role enacting agents, are in the place of *input.* There are three role enacting agents in this example, a Chinese company enacting the role of exporter, a Dutch company enacting the role of importer, and the Dutch Customs enacting the role of customs. Then the *initialization* transition will be unconditionally triggered and all the tokens are moved to the following places of *wait* according to the arc expressions.

Each place of *wait* specifies the color sets (role types) from its corresponding norm in the norm net and the arc expression related to the place selects the agents that match the color sets based on their roles. For example, the role of $norm_1$ is *exporter.* Therefore, the color set for the wait place in $norm_1$ and the related arc expression both try to match *exporter*, indicating that only agents enacting the role of exporter can move into this part of the net. Note that a token is only a reference to a role enacting agent and tokens representing the same agent can be in multiple places simultaneously, in the sense that the agent relates to a set of different norms. For example, the tokens representing the Dutch Customs are distributed to the places of both *$norm_2$* and *$norm_4$* since both norms involves the Dutch Customs. When the condition of a norm is satisfied and the agent in the place of *wait* performed the related action, the corresponding token will move to the state of *satisfied* or *violated* according to the type of the norm.

The *AND* (*OR*) relation between norms is mapped to *AND* transitions (*OR* transitions) in CPNs. However, for norms connected by *OE* operators, a special structure is used to indicate the exclusive and conditional relation between them. For example, the *violated* place of *$norm_3$* and an additional *wait* place are joined at an *AND* transition which is then connected to the *wait* place of *$norm_4$*, indicating that only when *$norm_3$* is violated can *$norm_4$* be triggered. In this way, the "conditional" part of the relation between the two norms connected by an *OE* operator is captured. For the "exclusive" part of the relation, we use *XOR* transition to connect the two norms, indicating that only one of the two norms can be satisfied. For instance, after the importer in the Netherlands presents the Dutch Customs with the specific origin documents, the state of *$norm_3$* changes to *satisfied* while *$norm_4$* has no chance to be triggered. However, when the action in *$norm_3$* is not performed before the required date, the state of *$norm_3$* changes to *violated* and at the same time *$norm_4$* is imposed as a sanction. Since the example is only a part of the EU regulations, the sanction to the violations of the norms in Figure 5 is not fully pictured. Finally, the corresponding CPN model ends with a token at the *output* place which indicates the compliance of a norm net instance.

Note that the mappings illustrated in this section is currently dedicated to a single norm net within a specific context. The mapping of hierarchical contextual norm nets is left as our future work.

5.3 Verification Properties

Based on the mappings from Norm Nets to CPNs, we can further explore the correspondences between the behavioral properties of Norm Nets and CPNs, so that the analysis techniques of CPNs can be utilized to facilitate the verification of the behavioral properties of Norm Nets. For example, the reachability property and liveness property in CPNs are interpreted as follows.

– **Reachability** indicates that, given a set of norms organized in a norm net, whether there is a possible way to comply with those norms, i.e., a path through the norm net (CPN) that is norm compliant at all steps. This property can be used to identify the inconsistencies between the norms.
– **Liveness** indicates that, given a set of norms organized in a norm net, whether some of the norms are never initiated, i.e., no occurrence sequences through the norm net (CPN) that includes those norms. This property can be used to identify the norms that are redundant or wrongly positioned.

Therefore, given a set of norms in MAS, we can first model them using the proposed normative structure, and then map the resulted norm nets to CPNs

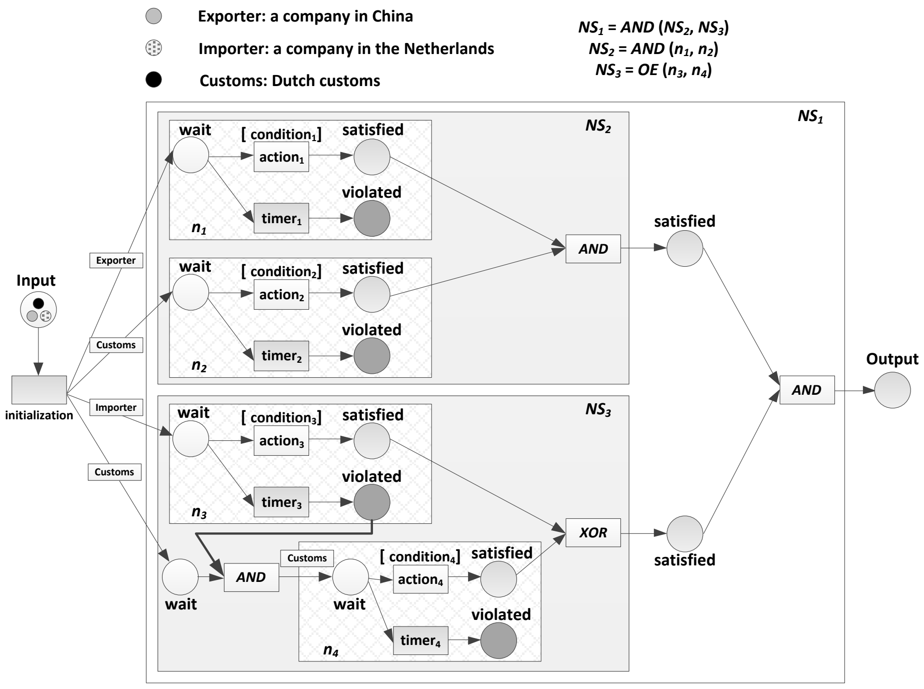


Fig. 5. Norm nets mapping to CPN

by which we can perform consistency and compliance checking on the norms. Furthermore, CPNs are fully supported by a number of analysis tools that can be used to implement such verifications [17].

6 Related Work

There is a growing interest in the research of norms to regulate and coordinate agents' behavior in MAS. Fruitful results have been achieved from different perspectives such as norm compliance, norm conflict resolution, norm contextualization, etc. [15] presented a formal normative framework intended to be used by agents that reason about why norms must be adopted and complied with, in which the relations between norms are represented as interlocking norms. The framework proposes that norms are applied in particular circumstances or within a specific context, but without considering the refinement relation of contexts, it is at a single level of analysis on norms. Munindar P. Singh proposed to use commitments to capture normative concepts in MAS and define norms as a tuple including subject, object, context, antecedent and consequent [18]. This approach provides a natural way to characterize the bounds of autonomy and interdependence between agents, but the contextual aspects of norms are not considered.

In [12], a formalism called Process Compliance Language (PCL) is proposed for the expression of violation conditions and the reparation obligations which is very important for formalizing norms to determine the compliance of a process with the relevant norms. PCL enables to represent exceptions as well as to capture violations and the obligations resulting from the violations, but it does not take a broader view on norm sets where relationships other than reparation of violation exist between norms. In order to regulate the behavior of agents in open and regulated MAS in a distributed manner, [11] presents a normative structure based on the propagation of normative positions as consequences of agents' actions and provides a mapping into Colored Petri Nets for conflict resolution. The normative structure models norms in normative scenes and builds connections between the scenes by transition rules, which focuses more on the causal relations between norms other than conjunction, disjunction and implication.

Since norms are usually specified at different levels of abstraction, there is a need to relate the abstract concepts used in the specification to concrete ones used in practice, which necessitates the research on norm contextualization. In [3], counts-as statements are used to provide the concrete concepts their institutional and organizational meaning according to different contexts and enable agents to reason about norm compliance by the context they are in. A context-based institutional normative environment is proposed in [5], which enables the use of norms within a hierarchical context structure and norm inheritance as a mechanism to facilitate contract establishment. Another perspective on contextual normative structure, presented in [10], models norms of MAS according to four levels of abstraction: Environment, Organization, Role and Interaction contexts. However, these contextual normative frameworks all concentrate on the effects of individual norms but ignore their relations.

Due to the changing nature of norms, conflicts or inconsistencies may occur. To solve this problem, different approaches have been proposed such as [19], [9], [4]. Eventhough, we do not focus on checking the consistency of normative structures, but since the building blocks (norm nets) are organized in such a way that they can be mapped to CPNs, inconsistencies will be easy to identified using CPNs formal analysis methods and tools.

7 Conclusions

In this paper, we proposed a normative structure that not only captures the characteristics of a single norm but also the relationships between norms. Given that agents in MASs interact with each other to achieve certain goals, the interrelated effects of norms on their behavior are very important for both individuals and the system. Therefore, the connections between norms should be explicitly indicated in a structural way. Moreover, contexts play an important role in the construction of norms, in the sense that the application of a norm heavily depends on its institutional context and a norm may have different interpretations in different situations. To this end, the concept of norm net in this paper expresses how a set of recursive norm sets organize in a hierarchy of contexts.

Most importantly, this paper presents a norm net contextualization process that describes norms from general to specific. This enables a modular approach for building normative structures that improves both reusability and flexibility. Furthermore, following this contextualization process, actors can have a better understand of their dos and don'ts with the evolution of contextual norm nets. To verify the proposal, we map norm nets to CPNs and incorporate agents/actors as colored tokens in the analysis, which presents the state transition process of norm nets and provides a potential approach for compliance checking on norms.

In future work, we intend to build a complete mapping for contextual norm nets from general to specific using advanced Colored Petri Nets, e.g., hierarchical CPNs. That is, linking the CPNs of abstract norm nets with that of contextual norm nets in a recursive manner and reflecting the contextualization process from the whole structure. Moreover, we would be interested in implementing simulation of norm nets for compliance checking on norms on the basis of CPN analysis tools and agent based simulation techniques.

References

1. http://ec.europa.eu/taxation_customs/customs/customs_duties/rules_origin/index_en.html
2. Aldewereld, H.: Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols. Utrecht University, PhD Thesis (2007)
3. Aldewereld, H., Álvarez-Napagao, S., Dignum, F., Vázquez-Salceda, J.: Making norms concrete. In: Proc. AAMAS (2010)
4. Boella, G., Pigozzi, G., van der Torre, L.: Normative framework for normative system change. In: Proc. AAMAS (2009)
5. Cardoso, H.L., Oliveira, E.: A Context-Based Institutional Normative Environment. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 140–155. Springer, Heidelberg (2009)
6. Criado, N., Argente, E., Botti, V., Noriega, P.: Reasoning about norm compliance. In: Proc. AAMAS (2011)
7. Dignum, F.: Abstract norms and electronic institutions. In: Proc. RASTA (2002)
8. Dignum, V., Dignum, F.: Can we ask you to read this paper? In: Proc. COIN (2007)
9. Esteva, M., Vasconcelos, W., Sierra, C., Rodríguez-Aguilar, J.A.: Norm Consistency in Electronic Institutions. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 494–505. Springer, Heidelberg (2004)
10. Felicíssimo, C., Choren, R., Briot, J., de Lucena, C.J.P., Chopinaud, C., El Fallah Seghrouchni, A.: Providing Contextual Norm Information in Open Multi-Agent Systems. In: Kolp, M., Henderson-Sellers, B., Mouratidis, H., Garcia, A., Ghose, A.K., Bresciani, P. (eds.) AOIS 2006. LNCS (LNAI), vol. 4898, pp. 19–36. Springer, Heidelberg (2008)
11. Gaertner, D., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J., Vasconcelos, W.: Distributed norm management in regulated multi-agent systems. In: Proc. AAMAS (2007)
12. Governatori, G., Rotolo, A.: How do agents comply with norms? In: Proc. WI-IAT. Dagstuhl Seminar Proceedings (2009)

13. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Uses. Springer (1997)
14. Kollingbaum, M.J., Vasconcelos, W.W., García-Camino, A., Norman, T.J.: Managing Conflict Resolution in Norm-Regulated Environments. In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G.A. (eds.) ESAW 2007. LNCS (LNAI), vol. 4995, pp. 55–71. Springer, Heidelberg (2008)
15. López y López, F., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. CMOT 12, 227–250 (2006)
16. Ostrom, E.: Understanding institutional diversity. Princeton University Press (2005)
17. Vinter Ratzer, A., Wells, L., Lassen, H.M., Laursen, M., Qvortrup, J.F., Stissing, M.S., Westergaard, M., Christensen, S., Jensen, K.: CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 450–462. Springer, Heidelberg (2003)
18. Singh, M.P.: Commitments in multiagent systems: Some history, some confusions, some controversies, some prospects. In: The Goals of Cognition. College Publications (2012)
19. Vasconcelos, W., Kollingbaum, M.J., Norman, T.J.: Resolving confict and inconsistency in norm-regulated virtual organizations. In: Proc. AAMAS (2007)
20. Zimmermann, A., Lorenz, A., Oppermann, R.: An Operational Definition of Context. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 558–571. Springer, Heidelberg (2007)

Programming Institutional Facts in Multi-Agent Systems

Maiquel de Brito[1], Jomi F. Hübner[1], and Rafael H. Bordini[2]

[1] Federal University of Santa Catarina
Florianpolis, SC, Brazil
{maiquel,jomi}@das.ufsc.br
[2] FACIN–PUCRS
Porto Alegre, RS, Brazil
r.bordini@pucrs.br

Abstract. In multi-agent systems with separate agents, environment, and institution dimensions, the institutional state can be affected by facts originating in any of those constituent dimensions. Most current approaches model the dynamics of the institution focusing on the agents and the institution itself as the main sources of facts that produce changes in the institutional state. In this paper, we investigate also the environment as an important source of facts that change the institution. We propose thus a model and a language to specify and program the institutional dynamics as consequence of events and state changes occurring in any of the three component dimensions of the system (agent, environment, and institution). Our approach was evaluated through a case study where we compare two solutions for an application: the original design and a new one based on our proposal. We observed a simplification of the agents' reasoning, an increase in the functions performed by the environment and the institution, and greater independence of the agents within the system. This last result is specially important in open systems where we cannot take for granted that agents will take part in the system.

Keywords: institutional facts, constitutive rules, environment, institution.

1 Introduction

This work considers a Multi-Agent System (MAS) as an open system with three distinct and independent dimensions: agents, institution, and environment. In open MAS, agents can enter or leave freely [3], and neither the number, nor the behaviour, nor the way in which the agents interact and access shared resources can be known at design time [11]. Therefore, an open MAS can have agent heterogeneity, conflicting individual goals, limited trust, and non-conformance to the specifications [2]. In order to conciliate the autonomy of the agents and the system goals, using an institutional dimension is a usual approach [9].

The institution can be affected directly by the actions of the agents (e.g. when an agent voluntarily adopts a role). In some cases, however, the institution can be affected by facts originating in the environment or even the institution. For instance, an agent running through a red traffic light is a fact essentially at the environment dimension. Although this fact is initially produced by an agent, it is also – or even *mainly* – an event of the environmental dimension. This fact means, in the institutional dimension, a norm violation. As we will illustrate in this paper, there are some advantages when considering the environment rather than the agents as the source of events that affect the institution. In order to model and implement this link between the institution and the other dimensions, some issues must be addressed. It is necessary to define what kind of facts can have an institutional consequence and how such facts can be checked for at the different dimensions.

The environment-based approach has been investigated in some related work (see Section 2.1). The work of Piunti [11] investigated this approach considering MAS with agents, environment, and institution as first-class abstractions. That work dealt with the question from a more conceptual perspective. In this paper, we continue the work of Piunti proposing a programming language to specify the dynamics of the institutional state as consequence of facts from both the environment and the institution[1]. The proposed language, presented in Section 3, assumes that such facts can be conceived as *count-as rules*. The underlying model (i.e. the Social Reality Theory of John Searle) and related work that deals with institutional facts in MAS are described in Section 2. The language implementation and evaluation is discussed in the context of a case study in Section 4. The main contributions of this work are: (i) a conceptual model of the dynamics of institutional facts that recognises the importance of the environment as one of the essential dimensions of multi-agent systems; (ii) a programming language to specify such dynamics which helps formalise the proposed model; and (iii) an implemented interpreter for the language thus making the work of practical use in multi-agent systems development.

2 Related Work

John Searle put forward the idea that reality is divided into brute and institutional portions [14]. The brute portion of reality is composed of elements that can be described by chemistry, physics, mathematics, etc. and does not depend on any beliefs or opinions from human beings. The institutional portion is composed of subjective elements and depends on collective agreements. Some facts occurring at the brute portion can have meaning at the institutional level. This meaning is stated by *constitutive rules* that have the form X *count as* Y *in* C where: (i) X is a *brute fact*; (ii) Y is an *institutional fact*, i.e. a fact at the institutional level that is a consequence of brute fact X; and (iii) C is the *context*

[1] We consider here that any action performed by agents has some effect in the environment and therefore, by considering the environment, we are also indirectly considering any fact produced as effect of the agents' actions.

where X *counts as* Y. For instance, a priest performing a ceremony (X) *counts as* an act of marriage (Y) if the act is performed in the correct way (C).

2.1 The Social Reality in MAS

Some related work, briefly analysed below, investigated how brute facts in MAS (agent interaction, agent actions in the environment, events occurred in the environment, etc.) may have meaning at the institutional level. They point to a correspondence, in MAS, to Searle's theory.

Table 1. Comparison of related models

Model	Dimensions			Brute facts			Language			Obj.
	Ag	Env	Inst	Act	Ev	St	Lang	Interp	App	
Artikis *et al.* [2]	✓		✓	✓			✓	✓	✓	✓
MASQ [15]	✓	✓			✓					
SEI [6]	✓		✓	✓						✓
Dastani *et al.* [8]	✓		✓			✓	✓			✓
Aldewereld *et al.* [1]	✓		✓	✓			✓	✓	✓	✓
Emb.Org. [11]	✓	✓	✓		✓		✓			✓

A first aspect to be considered in the models above is their authors' view about the dimensions of MAS. All models consider at least the agent dimension, and some of them focus on one particular dimension. The models proposed in [2,8,6,1] consider the existence of an institutional dimension but do not consider the environment as a first-class abstraction. In [2,1], the environment is conceived as being external to the MAS and institutional facts are triggered by agent actions. In [8,6], the environment is modelled as part of the institutional specification (rather than an independent dimension). More precisely, in [8] the institutional specification states the *effects* of the actions of the agents in the environmental elements, while in [6] the institutional specification has *modellers* and *staff* agents that are in charge of dealing with environmental entities.

A second aspect to be observed about the models is the source the brute facts, i.e. where they are produced. This aspect has a close relation to the dimensions that each model considers. Most of the models consider that agent actions are brute facts [2,6,1]. Events occurring in the environment are deemed as brute facts in [15] and [11]. The approach in [15] is subjective in the sense that *the agents* perceive and interpret the events. In a different way, the approach in [11] depends neither on agents' perceptions nor on their reasoning. That model considers that events occurring in the environment are brute facts regardless of the perception and reasoning of the agents. Another source of brute facts is the *state* of the system, as proposed in [8] and [1].

The third aspect that we have analysed is the existence of an implementation of the proposal. Although some models have an associated language, not all of the cited papers have described an interpreter or an application using the model and the language.

The last aspect considered in our analysis is where the count-as rules are processed. In the model presented in [15], count-as rules are independently evaluated by the agents based on their particular perception and knowledge. This evaluation is thus *subjective*, and each agent might have a different representation of the institutional state. In all other models, the evaluation is performed by something outside the agents and does not depend on them: it is thus an *objective* evaluation.

It can be noted that only two models consider the environment as a first-class abstraction that is a source of brute facts (Table 1). Among them, only in the model by Piunti et al. [11] the evaluation of brute facts is objective. Moreover, both models consider only events as brute facts and, as proposed in [8], a system state is a useful kind of brute fact. Although Piunti's approach includes a programming language, it does not describe an interpreter nor a practical application of the model. Our proposal, as presented in this paper, covers the following features: (i) considers the three distinct dimensions as first-class abstractions, (ii) the environment is considered as a source of brute facts, (iii) includes both events and state as leading to brute facts, (iv) is objective, and (v) has an implemented programming language.

3 Programming Institutional Facts

This section presents the proposed model and language that deal with changes in the institution as the result of events or states occurring in the environment or in the institution. Such changes are based on *count-as rules*.

We assume that the environment and the institution states are not fully accessible. Thus, among the elements that compose those dimensions, there is an *observable* portion that is considered in our model. About the unobservable portion, its existence is assumed but is not of our concern. Figure 1 illustrates this approach for the environment and the institution dimensions. Each dimension has a state and events of which a portion is observable (the grey portions). The arrows represent the direction of the count-as rules: particular events and states in the environment/institution can produce changes in the institutional state.

Definition 1 (Observable event). *An event is an instantaneous and abrupt happening occurring inside or outside a system. These occurrences can be triggered, for instance, by an agent or by the environment. An event can have, as consequence, some change in the system state. Such change, however, is not mandatory and, if it happens, does not need to be perceivable by an external observer [7]. We define E_e to be the set of all observable events in the environment and E_i the set of all observable events in the institution. Events are represented as predicates.*

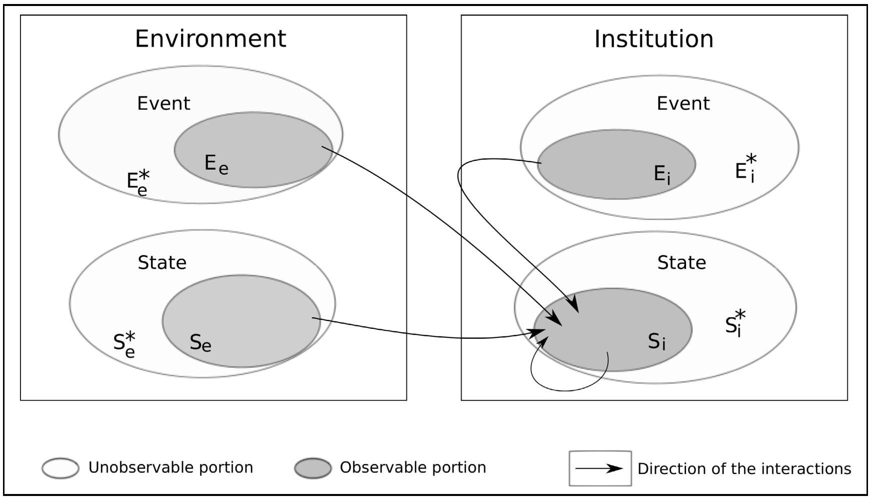


Fig. 1. Observable and unobservable portions of environment and institution

Definition 2 (Observable state). *Let P_e be the set of all observable properties, represented by predicates, of the environment. An observable state of the environment s_e is a subset of P_e. The set of all possible states of the environment is represented by $S_e = 2^{P_e}$.*

Similarly, let P_i be the set of all observable properties of the institution. An observable state of the institution s_i is a subset of P_i. The set of all possible states of the institution is represented by $S_i = 2^{P_i}$.

Definition 3 (Domain Knowledge Base). *A Domain-Knowledge Statement (DKS) is a predicate representing some knowledge that belongs to the institutional dimension of a particular application. A set of domain-knowledge statements is a Domain Knowledge Base (DKB).*

For instance, in a university scenario, an agent entering a classroom *count as* this agent being a student. The environment of the university can have several rooms and it is out of the environment specification to define the institutional meaning of the rooms (classrooms, teachers room, conference room, etc.). The DKB can be used, for example, to state that some room is a classroom.

Having an explicit DKB part of a program also makes it easier to write count-as rules, as they typically become more compact and readable. For example, by having in the DKB the statements $is\_classroom(room210)$ and $is\_classroom$ $(room440)$, we can write a count-as rule to the effect that "entering a *classroom* count-as adopting the role of student" without having to write rules for each individual room. Furthermore, an agent entering room 210 or 440 could count as adopting a role whilst entering a conference room in the same building would have no institutional meaning whatsoever.

3.1 Programming Language

With the elements previously defined, we introduce a language to program *count-as rules* in a *count-as program*. The syntax of the language is given in Figure 2.

A *count-as program* is composed of (i) a DKB, i.e. a set of domain-knowledge statements, and (ii) a set of *count-as rules*. The count-as rules are the main part of the program as they allow users to define the institutional dynamics of their multi-agent applications following our approach. They are explained in detail below.

```
count_as_program  ::= (dkb)? count_as_rule+
dkb               ::= 'domain_knowledge_base:' (predicate '.')+
count_as_rule     ::= termX 'count-as' termY ('in' context)? '.'
termX             ::= event | state
event             ::= '+' predicate
state             ::= '*' formula
termY             ::= predicate (',' predicate)*
context           ::= formula
```

Fig. 2. Grammar of the proposed language

Besides the grammar for a count-as program, we also give the following definition that will help the formalisation we provide later in this section.

Definition 4 (Count-as program). *A* count-as program *(corresponding to* `count_as_program` *in the grammar) is a tuple $\langle R_e, R_s, D_k \rangle$ where (i) R_e is a set of count-as rules that deal with events (*event-count-as rules*), (ii) R_s is a set of count-as rules that deal with state (*state-count-as rules*), and (iii) D_k is a set of* DKB *statements.*

3.2 Count-as Rules

A count-as rule (element `count_as_rule` of the grammar) is inspired by the idea of *constitutive rules* put forward by John Searle and have the form *X count-as Y in C*. The *X* term may be an event or a state. We define thus two kinds of rules: *event-count-as* to deal with events and *state-count-as* to deal with state. Both can also be formalised as below:

Definition 5 (Event-count-as rule).
An event-count-as rule is a tuple $\langle b_f, i_f, c \rangle$ where:

- *$b_f \in E_e \cup E_i$ is an event that led to a brute fact;*
- *$i_f \in 2^{P_i}$ is a set of institutional properties that become true of the institution through the application of the rule;*
- *c is a logical formula composed of predicates belonging to P_e and P_i which point to the observable state of the environment and institution; the formula must be true for the rule to apply.*

An *event-count-as* rule defines a new institutional state i_f as consequence of the occurrence of event b_f in a context c. Suppose that an agent getting into a classroom at 10am on a Friday counts as this agent playing the student role. This is an example of an *event-count-as* rule where: (i) the event of the agent getting into the classroom is the brute fact b_f, (ii) the institutional fact of the agent playing the student role is the consequence of the rule application (i_f), and (iii) the time when the event happens is the context where the rule is applicable (c). Figure 3 (left) shows an example of this type of rule. Notice that the event of leaving the classroom does not cancels the effect of the count-as rule. If that was meant to be the case, a new count-as rule could be written, stating that an event triggered when an agent leaves the classroom count-as the agent leaving the student role.

```
+ getInTheRoom(101)[agent(Agent)]
count-as
  play(Ag, student)
in
  time("10 am") &
  day(friday).
```

```
* studentsInTheClassroom(101,X) & X >= 30 &
  agentInTheRoom(Agent,101) &
  play(Agent, teacher)
count-as
  classStarted(artificialIntelligence)
in
  time("10 am") & day(friday).
```

Fig. 3. *Event-count-as rule* (left) and *State-count-as rule* (right)

Definition 6 (State-count-as rule). *A state count-as rule is a tuple* $\langle b_f, i_f, c\rangle$ *where:*

- b_f *is a logical formula composed of predicates belonging to* P_e *and* P_i *which point to the observable state of the environment and institution; the formula must be true for the rule to apply;*
- $i_f \in 2^{P_i}$ *is a set of institutional properties that become true of the institution through the application of the rule;*
- c *is a logical formula composed of predicates belonging to* P_e *and* P_i *which point to the observable state of the environment and institution; the formula must be true for the rule to apply.*

A *state-count-as* rule defines a new institutional state (given by the properties in i_f) as consequence of the current *state* of environment and institution. Suppose that in a university scenario, if there are more than 30 students and a teacher in a classroom on Friday at 10am, it means that the class has started (see Figure 3 (right)). This is an example of a *state-count-as* rule where: (i) the brute fact b_f is the conjunction of the environmental property of some agents being in the classroom and the institutional property of these agents playing the roles of student and teacher, (ii) the institutional property of the class having started is the consequence of the rule application, and (iii) the time when those properties hold is the context where the rule is applicable (c). Notice that the effect of the count-as rule is not cancelled when the state b_f ceases to hold. In this case, a new count-as rule could be written to explicitly define a new institutional state when b_f is not true.

The reasons and advantages of having these two kinds of count-as rules are discussed in Section 4.3.

3.3 Language Semantics

In this section the semantics of the language is formalised using structural operational semantics [12]. The interpreter for the count-as program is placed side by side with environmental and institutional platforms. It is constantly informed by these platforms about successful events[2] and new states and, as the result of the application of some count-as rule, the interpreter sends to the institutional platform what should be its next state. Notice that we consider multi-agent systems where there is only one institution and one environment model, and each runs on a single host. Issues related to distribution and topology are beyond the scope of this paper.

The operational semantics is given as a transition system where a particular state of the system is represented by a *configuration* as formally defined below.

Definition 7. *The transition system configuration is a tuple* $\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle$*, where:*

- R_e *is a set of* event-count-as *rules provided by (the parsing of) the count-as program;*
- R_s *is a set of* state-count-as *rules provided by the count-as program;*
- D *is a set of* DKB *statements also provided by the count-as program;*
- $\mathcal{E}$ *is a queue of events* $e \in E_e \cup E_i$ *provided by the environment and institution platforms;*
- $\mathcal{N}$ *is a set of predicates representing the observable state of the environment as provided by the environment platform;*
- $\mathcal{I}$ *is a set of predicates representing the observable state of the institution as provided by the institution platform;*
- $\mathcal{T}$ *is a queue of properties that are the result of the interpretation of the rules and must become true of the institution.*

The initial configuration is $\langle R_e, R_s, D, \emptyset, \emptyset, \emptyset, \emptyset\rangle$. As the interpreter runs, and events and states are informed by the platforms, this configuration evolves as defined by the following transition rules. Due to the lack of space, we will explain only the main transitions of the semantics and omit the transition rules related to addition and deletion of count-as rules during the execution of the count-as program.

Event Processing

Let $\mathsf{head}(\mathcal{E})$ be a function that returns the head of a list, $\mathsf{tail}(\mathcal{E})$ be a function that returns the tail of a list, and θ be a substitution of all variables of the brute

[2] We assume that the reported events represent the consequence of successful action in the environment; when attempted actions fail we assume that no event is generated or at least that they are not reported to our interpreter.

fact in the rule. If there is an *event-count-as* rule where $b_f\theta$ is equal to the event given by $\mathsf{head}(\mathcal{E})$, the term c is a logical consequence of the state of environment and institution, and the count-as consequence i_f does not belong to the current state of the institution, then the rule fires. As a result, the properties expressed by i_f will be added to the result queue $\mathcal{T}$.

$$\frac{\langle b_f, i_f, c\rangle \in R_e \qquad b_f\theta = \mathsf{head}(\mathcal{E}) \qquad \mathcal{N} \cup \mathcal{I} \cup D \models c \qquad i_f \notin \mathcal{I}}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle \longrightarrow \langle R_e, R_s, D, \mathsf{tail}(\mathcal{E}), \mathcal{N}, \mathcal{I}, \mathcal{T} \cup i_f\rangle}$$

State Processing

Each *state-count-as* rule $r_s \in R_s$ whose brute fact b_f and context c are logical consequences of the state of the environment and institution is triggered and its properties expressed by i_f are added to the result queue $\mathcal{T}$.

$$\frac{\langle b_f, i_f, c\rangle \in R_s \qquad \mathcal{N} \cup \mathcal{I} \cup D \models b_f \qquad \mathcal{N} \cup \mathcal{I} \cup D \models c \qquad i_f \notin \mathcal{I}}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle \longrightarrow \langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \cup i_f\rangle}$$

Passing the Resulting Properties to the Institution

If $\mathcal{T} \neq \emptyset$, the institution platform *pt* consumes the first property from queue $\mathcal{T}$ and changes the institutional state accordingly.

$$\frac{\mathcal{T} \neq \emptyset \qquad t = \mathsf{head}(\mathcal{T}) \qquad \mathsf{consume}_{pt}(t)}{\langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T}\rangle \longrightarrow \langle R_e, R_s, D, \mathcal{E}, \mathcal{N}, \mathcal{I}, \mathcal{T} \setminus t\rangle}$$

4 Case Study

In order to evaluate the proposal, we implemented the language interpreter and its interface with the environmental and institutional platforms of the JaCaMo framework[3]. Our count-as language was used to develop a new version of the *Build-a-House* example [4]. This example is suitable for our evaluation since it was originally designed with the three dimensions (agents, institution, and environment) in mind, as the platform allows explicit programming of all three levels. The agents in JaCaMo are programmed in *Jason* [5], the environment is programmed in CArtAgO [13], and the institution is programmed in $\mathcal{M}$oise [10].

4.1 Original Implementation

The example concerns a multi-agent system representing the inter-organisational workflow involved in the construction of a house. An agent called Giacomo owns a plot and wants to build a house on it. In order to achieve this overall goal, first

[3] Due to the lack of space, the details of these implementations are not described in this paper.

Giacomo will have to hire various specialised companies (the *contracting phase*) and then ensure that the contractors coordinate and execute the various required tasks required to build a house (the *building phase*). For each company, there is a *company* agent participating in the contracting phase and then, possibly, in the building phase too.

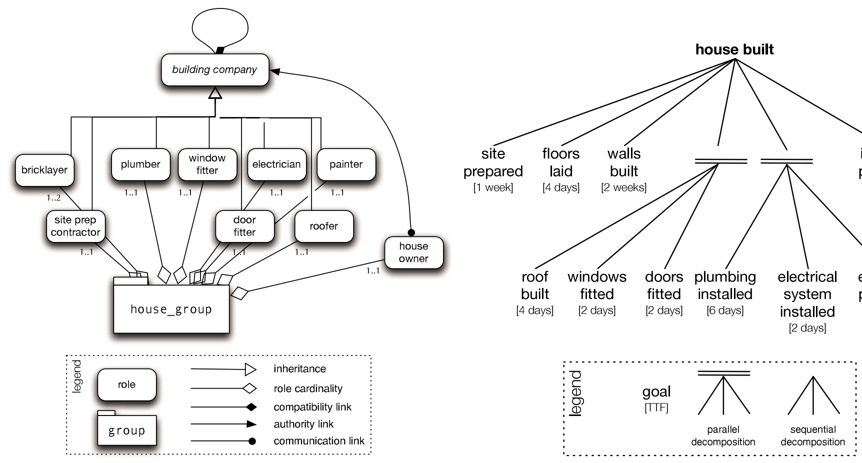
In the contracting phase, Giacomo starts one auction for each of the several tasks involved on the building of the house, such as site preparation, laying floors, building walls, etc. The auction starts with the maximum price that Giacomo can pay for a given task and companies that can do the task may offer a price lower than the current bid. After a given deadline (known by Giacomo but unknown to the bidders), for each auction Giacomo: (i) stops it, (ii) checks which company proposed the lowest price, and (iii) sends a message to that company hiring it.

After the companies have been hired, the contractors have to execute their tasks in a coordinate way during the building phase. Each company has to join the organisation adopting a specific role and, by doing so, it becomes responsible for some goals in the overall process of building the house. The organisation is specified in Figure 4 using the $\mathcal{M}$oise notation. The structural specification defines a group where company agents will play the sub-role `building company` and the Giacomo agent plays the role `house owner`. The functional specification decomposes the organisational goals into sub-goals, defines the sequence in which each will be achieved and gives a "time-to-fulfil" (TTF), i.e. a deadline, for each sub-goal. The normative specification determines which goals an agent playing a given role is obliged to achieve. Thus, the agents must be attentive to the organisation in order to know what are their obligations. By perceiving that new obligations are in place for themselves, the agents can trigger the execution of particular plans in order to achieve the organisational goals and then inform the goal achievement back to the organisation.

4.2 Implementation with Count-as Rules

The new implementation of the *Build-a-House* example, using count-as rules and a DKB, allows the change of the institutional state as a result of facts about the environment and in the institution. Several count-as rules can be defined for this example; here, however, we will explain only the most illustrative rules.

For instance, in the original example Giacomo needs to (i) control the deadline of the auctions, (ii) check which agent is the winner of every auction, and (iii) send a message to these winners asking them to adopt the corresponding role in the organisation. In the new implementation, thanks to the count-as program, Giacomo only needs to control the deadline and finish the auctions, and company agents do not need to explicitly adopt roles. The count-as rule in Figure 5 handles the adoption of roles for the companies. When the auction state is closed (for instance, by a Giacomo action or some deadline), the current winner will automatically start playing the corresponding role (and it will be informed of that by the organisation). In that rule, `auctionStatus` and `currentWinner` are properties provided by the environment platform, `play` is the property that has to become true of the institution, and `auction_role` is a DKS.



norm	role	mission / goals
n1	house owner	house built
n2	site prep contractor	site prepared
n3	bricklayer	floors laid, walls built
...		

Fig. 4. Organisational specification of example *Build-a-House*: structural specification (left), functional specification (right) and normative specification (bottom) [4]

```
/* If an auction "Art" is finished, its winner ("Winner") plays a role "Role",
   if it doesn't adopted it yet  */
* auctionStatus(closed)[source(Art)]
count-as
  play(Winner,Role,hsh_group)[source(hsh_group)]
in
  currentWinner(Winner)[source(Art)] & not(Winner==no_winner) &
  auction_role(Art,Role).
```

Fig. 5. Count-as rules for role adoption

In the original implementation, for each goal related to the house building, company agents execute operations on the environment that simulate the real task. Besides the execution of those operations, the agents must inform the organisation about the achievement of organisational goals. In the new implementation, these executions count-as the achievement of organisational goals. Thus, the agents need only to act on the environment and the achievement of the goals is informed to the organisation by the count-as interpreter. Figure 6 shows an example of a rule stating that the occurrence of the event `prepareSite` (which is the result of an operation on the environment simulator) count-as the achievement of the organisational goal `site_prepared` (this goal is defined in the functional specification illustrated in Figure 4).

```
/* The occurrence of the event "prepareSite" means
   the achievement of organisational goal "site_prepared" */
+ prepareSite[agent_name(Ag),artifact_name(housegui)]
count-as
  goalState(bhsch,site_prepared,Ag,Ag,satisfied)[source(bhsch)].
```

Fig. 6. Count-as rules for organisational goal achievement

We defined two sets of DKS. The first one defines the roles given to the winners of each auction. The second one defines the missions (i.e. sets of goals) attributed to agents that are playing specific roles. Figure 7 shows some examples of such statements.

```
domain_knowledge_base:
   /* role_mission(R, M): Relates a role "R" to a mission "M" */
   role_mission(electrician,install_electrical_system).
   role_mission(painter,paint_house).

   /* auction_role(A, R): Relates an auction "A" to a role "R" */
   auction_role(auction_for_ElectricalSystem, electrician).
   auction_role(auction_for_Painting,         painter).
```

Fig. 7. Domain knowledge statements

4.3 Case Study Discussion

The use of count-as rules has three initial advantages. The first is that agents do not need to be aware of the organisation or even to reason about it, unless that makes sense in the particular application. In the new version of the building a house scenario, company agents do not need to adopt roles, reason about their roles, etc. Trivial role adoption can be done by the count-as interpreter based on brute facts caused by the companies. The second advantage is a consequence of the first: agents cannot avoid the institutional consequences of their actions either (which in some application might be very important, particularly in open system). In the original implementation, Giacomo asks the companies to adopt the corresponding roles when they win the auction. However, the companies can simply ignore the request and do not adopt the role (as they ought to in this application). More importantly, if a company actually prepares the site but does not tell the organisation, the institution simply becomes inconsistent, and as a consequence the system would simply halt waiting for something that already happened.

One can argue that we are limiting the autonomy of the agents using this kind of count-as rules. However, the motivation for this approach is precisely to handle the autonomy of the agents in open systems, where some restrain on agents' autonomy is required anyway. Moreover, the designer of the system may include or not this kind of count-as rules depending on the requirements of the application. In some cases, the count-as rules do not mean less autonomy than without them, it only means more readable code and conceptually more adequate declarative representations.

The third advantage of the proposal is precisely the simplification of the reasoning and action of the agents and the agent programs. Due to the possibility of modelling institutional consequences based on events and states, agents do not need to perform some actions on the institution. For example, the agent Giacomo performs 46 actions in the original example while this number is reduced to 19 in the new implementation. Table 2 lists the main activities of an agent named *CompanyA* in both the original example and in our experiment. The number of different tasks performed by CompanyA in the original example is 8 while in the new implementation this number is reduced to 4. This reduction does not necessarily mean, however, either an improvement on system performance or a reduction in coding. It is essentially a conceptual change, as part of the code was moved from the agents program to the count-as rules. That moved code is better conceived as belonging to the institution than to the agents, so it is more coherent to program it outside of and independently from the agents. Our approach therefore appears to further improve the programming style available in a multi-agent oriented programming platform where the three distinct dimensions of a multi-agent system can be directly programmed.

Besides the simplification of the agents' reasoning and action, we noticed an improvement of the institutional dimension in the system that was implemented following our approach. The institutional dimension is composed of various kinds of mechanisms with the aim of keeping the system in a consistent state despite its openness and the agents' autonomy. We regard count-as rules as playing an important part related to this aim. As illustrated by the two examples of count-as rules given above, it is possible to claim that the count-as rules are a mechanism that give institutional meaning to events and states of the environment and the institution. This meaning typically does not depend on agents participating in a particular episode of an institution.

As described in Section 2, while some authors prefer to use events as brute facts, others prefer states. In our point of view both approaches are useful and were thus included in our proposal. We point out three main reason for our decision:

- **Partial Knowledge about the Institutional and Environmental Models.** We assume the possibility of incomplete knowledge about the institutional and environmental models we are dealing with. It is possible then that a system designer does not know all the events that produce some particular state, and they may thus prefer to write count-as rules using states. Conversely, designers may not know the complete system states generated after some relevant events, so they may prefer to write count-as rule using events as triggers instead.

Table 2. Activities of agent CompanyA

	Original example	New implementation
Look for the group	✓	✓
Look for auctions	✓	✓
Submit bids to auctions	✓	✓
Receive the contracting message	✓	
Adopt a role	✓	
Commitment to a mission corresponding to the adopted role	✓	
Execute plans related to the mission	✓	✓
Inform the organisation about goal achievement	✓	
Total	8	4

- **Expressiveness of Count-as Rules.** In particular cases, several events can produce the same state of interest. Thus, a single *state-count-as* rule can replace several *event-count-as* rules. Similarly, an event of interest can happen in various different states, so a single *event-count-as* rule might suffice to cover various *state-count-as*, depending on the circumstances.
- **Concurrency and Ordering of Events.** In the example of the classroom, suppose a scenario where a teacher entering into the classroom counts as the class having started if there is at least one student in the classroom. In the case where the teacher enters into the classroom and there is no student, the rule is not triggered. However, as soon as a student enters in the room, the class should be considered as started. Another event-count-as rule is then needed (triggered by the student entering the room). Where various events lead to particular circumstances and the order does not matter, typically a state-based representation might be more useful.

  As mentioned above, the *event-count-as* rules can be more suitable or intuitive to programmers. Additionally, when evaluating rule application, matching events seems to be faster than evaluation of an overall state (we aim to evaluate this experimentally in future work).

Although both kinds of rules are useful depending on the application domain, if we know all the institutional and environmental models of some application, any event-count-as can be rewritten as state-count-as (assuming that every event produce a change in the state) and vice-versa (assuming that every state change is produced by some event).

5 Conclusions

In this paper, we proposed a model and programming language for specifying the institutional dynamics in multi-agent systems that are based on three distinct dimensions (i.e. agents, environment, and institution). An important feature of our approach is that we consider both events and states of environment and institution as brute facts. The contributions of this work include a programming language and its interpreter that allowed us, in the case study presented here, to simplify the programming of an application and make it more robust against malevolent agents in open systems. In future work, we plan to evaluate the performance of this interpreter and particularly the two types of count-as rules available in our approach. We also plan to deal with issues related to topology and distrubution.

Acknowledgments. The authors are grateful for the support given by CNPq, grants 307924/2009-2, 307350/2009-6, and 478780/2009-5, and by CAPES. We would like to thank Alessandro Ricci, Michele Piunti, and Olivier Boissier for their contributions to this work.

References

1. Aldewereld, H., Alvares-Napagao, S., Dignum, F., Vasquez-Salceda, J.: Making norms concrete. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), vol. 1, pp. 807–814. International Foundation for Autonomous Agents and Multiagent Systems, Toronto (2010)
2. Artikis, A., Pitt, J., Sergot, M.: Animated specifications of computational societies. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3 (AAMAS 2002), pp. 1053–1061. ACM, New York (2002)
3. Boissier, O., Hübner, J.F., Sichman, J.S.: Organization Oriented Programming: From Closed to Open Organizations. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 86–105. Springer, Heidelberg (2007)
4. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. In: Science of Computer Programming (2011)
5. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. In: Wiley Series in Agent Technology. John Wiley & Sons (2007)
6. Campos, J., López-Sánchez, M., Rodríguez-Aguilar, J.A., Esteva, M.: Formalising Situatedness and Adaptation in Electronic Institutions. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 126–139. Springer, Heidelberg (2009)
7. Cassandras, C.G., Lafortune, S.: Introduction to discrete event systems. Springer (2008)

8. Dastani, M., Tinnemeier, N., Meyer, J.C.: A programming language for normative multi-agent systems. In: Dignum, V. (ed.) Multi-Agent Systems: Semantics and Dynamics of Organizational Models. Cap. XVI, pp. 397–417. Information Science Reference, Hershey (2009)
9. Esteva, M., Rosell, B., Rodriguez-Aguilar, J.A., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Jennings, N.R., Sierra, C., Sonenberg, L., Tambe, M. (eds.) Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), vol. 1, pp. 236–243. ACM, Washington, DC (2004)
10. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multi-agent systems using the MOISE+ model: programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering 1(3/4), 370–395 (2007)
11. Piunti, M.: Situating agents and organisations in artifact-based work environments. PhD Thesis, Univerist di Bologna (2009)
12. Plotkin, G.D.: A structural approach to operational semantics. J. Log. Algebr. Program. 60-61, 17–139 (2004)
13. Ricci, A., Piunti, M., Viroli, M., Omicini, A.: Environment Programming in CArtAgO. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 259–288. Springer (2009)
14. Searle, J.: The construction of social reality. Free Press (1999)
15. Stratulat, T., Ferber, J., Tranier, J.: MASQ: Towards an integral approach to interaction. In: Proceedings of the 8th Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Richland, SC, vol. 2, pp. 813–820 (2009)

Towards a General Model for Adapting Structure while Maintaining Topology: Pipelines

Matthew Shaw, Jeroen Keppens, Michael Luck, and Simon Miles

Department of Informatics, King's College London, Strand, London, UK
matthew.shaw@kcl.ac.uk

Abstract. Large scale information systems are increasingly structured around flexible workflows of services providing a range of functionalities that are configured to suit particular needs, yet this flexibility can bring a lack of organisation in the ways in which services are combined. Particular system structures bring different benefits to an application in terms of efficacy and efficiency but sometimes need to reorganise as their circumstances change. In this context, this paper seeks to establish techniques for reorganisation that preserve particular topologies in support of their recognised benefit for the target applications. The contributions are twofold: first, a general vision of reorganisation of defined topologies, in which topology is preserved but efficiency and efficacy are optimised; and second, a specific solution for the case of pipelines, reorganising to optimise key application-specific metrics, while preserving topology. The paper is thus the starting point for a more ambitious general programme of research.

Keywords: Self-organisation, reorganisation, adaptation, topology, organisational structure, pipelines.

1 Introduction

Large scale information systems are increasingly structured around workflows of services providing a range of functionalities that are configured to suit particular needs. Indeed, the construction of combinations of services to form systems satisfying application-specific demands offers a flexibility that is often missing in rigid system structures. However, this flexibility can bring a lack of organisation in the ways in which services are combined. For example, many applications are naturally hierarchical in nature and are thus best suited to a hierarchical organisation of services. Other applications will suggest alternative structural arrangements, some of which may correspond to well-known organisations, and some of which may be ad hoc instead.

Now, the suitability of an organisational structure to an application in this sense lies in the efficacy and efficiency of the structure in supporting the particular goals of the system. This may be in terms of minimising load, in maximising throughput, or in other such objectives to be optimised. This paper recognises the value of such organisational structures yet seeks to provide a means of enabling them to reorganise as their circumstances change. Indeed reorganisation is known to be a valuable technique in the armoury of modern computing systems, motivated in part by increasing interest in areas such as autonomic computing. Importantly, however, rather than allowing arbitrary ad hoc structures to emerge, this paper takes a different standpoint in seeking to establish techniques for reorganisation that preserve particular topologies in support of their recognised benefit for the target applications.

As indicated above, our work is concerned with the key concepts of *structure* and *topology*, but these these terms are unfortunately often conflated. In some work, the term structure is used to represent the collection of connections or interrelationships between agents in a system [14], while in other work it is used in a more general sense, encompassing roles and social norms, as well as interaction constraints [3]. For the purpose of clarity, in this paper, we take an organisation's *structure* to be nothing more than a configuration of connections or relationships between all agents within an organisation. More specifically, a structure can be understood as a set of agent pairs, where each pair represents some connection between them. Now, an organisation's *structure* is often constrained by a particular *topology*. In this sense, a topology represents some pattern, or rule, that constrains the connections allowed in an organisation's structure, thereby producing structures that follow topologies, such as pipelines, or hierarchies. In this paper, we adopt this terminology, and use it throughout.

The contributions of this paper are twofold. First, the paper presents a general vision of reorganisation of defined topologies, in which topology is preserved, but efficiency and efficacy are optimised. Ultimately, this will lead to a library of techniques for topology-preserving reorganisation across different topologies, and potentially a means to transform between topologies as needs demand. Second, the paper instantiates this broad vision with a specific solution for the case of pipelines, reorganising to optimise key application-specific metrics, while preserving topology. It is thus the starting point for the more ambitious general programme of research.

The paper is structured as follows. The next section motivates the paper as whole by presenting a motivating scenario and a task allocation model setting out the problem we address. Then, in Section 3, we describe the main contribution of the paper, the techniques for reorganising, both in general and in the specific case of pipelines. Section 4 presents the initial results obtained with our techniques, before reviewing related work in Section 5, and concluding in Section 6.

2 Task Allocation and Execution Model

2.1 eScience Scenario

To motivate our work, we introduce a scenario based in the domain of eScience. Consider a large, potentially global network of electronic resources (such as devices, or even data) that are owned by different institutions, all willing to pool their individual resources in order to gain access to a larger set of shared resources that would otherwise be unavailable to them. All resources are networked, with some performing computationally intensive tasks like the analysis of large scientific data sets such as resulting from the Large Hadron Collider (LHC) at CERN [7,8]. Processing this data takes considerable time, and we want to process it as quickly as possible. If there is only one task that cannot be processed concurrently across multiple machines, then the optimal solution is for it to be processed on the fastest machine.

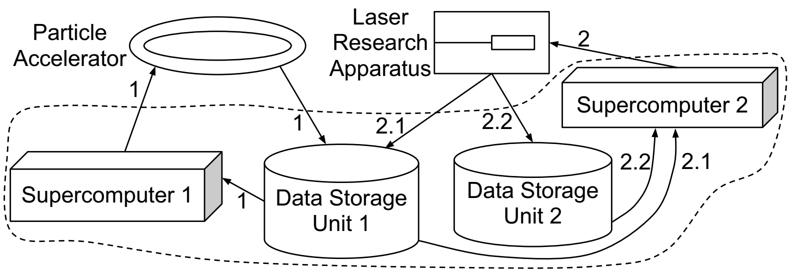


Fig. 1. Visualisation of the eScience scenario

Now, suppose there are two research centres, each using a *particle accelerator* (PA), and a *laser research apparatus* (LRA) respectively, and both generate large data sets. Such data needs to be stored and then processed, but neither research facility has the ability to do so. However, there are two data storage units, (DSU1 and DSU2), that are each capable of storing 100 units of data, and two supercomputers, (SC1 and SC2), that are capable of processing data, with SC1 processing data faster than SC2. This is summarised in Table 1.[1]

Table 1. The capabilities and resources available in the eScience scenario

		Computational Devices					
		SC1	SC2	DSU1	DSU2	PA	LRA
Services	Data Storage	N	N	Y	Y	N	N
	Processing Power	Y	Y	N	N	N	N
Hardware Specs	Memory	-	-	100	100	-	-
	Processing Power	Faster	Slower	-	-	-	-

If PA and LRA simultaneously perform experiments, respectively producing 50 units and 120 units of data, PA can store its data at DSU1, but LRA stores 100 units at DSU2 and 20 units at DSU1, since neither DSU has the capacity for 120 units. PA's data is then passed to the fastest supercomputer, SC1, for processing and, since SC1 is now busy, LRA's data is passed to SC2 despite it being slower. This is illustrated in Figure 1, in which the arrows labelled 1 show PA's task being completed, the arrow labelled 2 shows LRA's task being completed, and arrows labelled 2.1 and 2.2 show the task being decomposed.

2.2 Task and Agent Model

In this scenario we require the different computational entities to undertake various tasks (storing data or processing it), and to pass these tasks to others if the entities

[1] The capabilities of PA and LRA are not relevant, so are omitted.

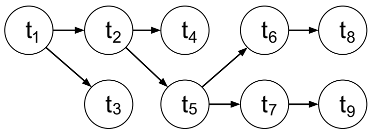


Fig. 2. Example of a task hierarchy

themselves cannot execute them. The key *task* of this scenario is to analyse data. In this sense, a task satisfies a particular *requirement*, where that requirement amounts to a specification of the *services* needed to perform that task, and the time for which each such service is needed. Tasks may also be decomposed into subtasks (potentially with ordering constraints): for example, to analyse data it must first be stored, then processed.

In our example, DSU1 and SC1 offer *services* for processing and storing data. The details of such services are unimportant for our purposes, and we simply specify the set of all services, $S = \{s_1, s_2, \dots\}$. Clearly, in order to fulfil tasks, services must perform some work. A *requirement* is a specification of the services and the amount of work needed from each in order to achieve the task. For simplicity, we assume that all services provide the same amount of work, or effort, per unit of time, and we use time as a simple proxy for an amount of work. In this way, a service may be required for 3 units of time, while another is required for 6 units of time. A requirement r thus takes the form $(s, reqt)$, where $s \in S$ is the required service, and $reqt \in \mathbb{Z}+$ is the amount of time for which it is required. The enactment of a service to satisfy a task's requirement is encapsulated as a *service instance*, in the form, $si = (t, s)$, where s is the service satisfying task t. Since tasks are decomposable, as indicated above, they are represented in a tree structure, as in Figure 2, where each vertex indicates a subtask, and each line an ordering constraint between subtasks. These constraints mean that t_1 must be completed before t_2 and t_3 can begin, but once t_1 is completed, t_2 and t_3 can be executed concurrently. To represent this additional complexity, a task takes the form $t = (r, \text{SUBT})$ where r is a requirement, and SUBT is a set of subtasks.

Given all this, devices such as PA or DSU1 are represented as *agents*, that use their services SERV to execute tasks that are in its list of tasks, TASKS. Each agent has a capacity $cap \in \mathbb{Z}+$ limiting the number of service instances it can run concurrently, to capture limited resources such as memory in a data storage unit. As each service is used, a service instance is created and added to an agent's set of current service instances SI, such that $|\text{SI}| \leq cap$. Once all of a task's requirements have been met, the service instance will be removed, allowing more to be created. Finally, an agent has a set CON of connections with other agents. An agent a, therefore, is represented as $a = (\text{SERV}, \text{TASKS}, cap, \text{SI}, \text{CON})$.

While agents are clearly by definition autonomous and social, with the ability to refuse to undertake tasks delegated to them, for example, our focus in this paper is not on the internal decision-making apparatus of agents, but instead on the macro-level of organisation. For this reason, and to ensure clarity and simplicity of presentation, in this paper we abstract away from any details of autonomy and decision-making, and assume that agents are benevolent, executing any tasks they are allocated. Clearly, more sophisticated models are possible, but that is outside the scope of the work presented here.

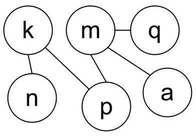


Fig. 3. An example of an organisational structure represented as a graph

2.3 Task Allocation

Given our basic model above, we can now consider how agents are allocated tasks. We adopt a simple model with the assumption that time is in discrete units, with a number of rounds, in each of which an agent a undertakes two major activities: it manages its TASKS list; and it executes tasks. In managing its tasks, *a* first places any received tasks in its TASKS list. Then, in order of arrival, each task *t* in the list is reviewed: if *t*'s requirements can be satisfied directly by *a*, and a has capacity to do so, a creates a service instance *si* to execute *t*, adds *si* to SI, and removes *t* from the TASKS list; if *a* can satisfy *t*'s requirements, but does not currently have capacity to do so, then the task remains on the list, waiting for capacity to become available; finally, if *a* cannot satisfy *t*'s requirements, then it must find another agent to which to delegate the task.

Once *a* has finished managing its TASKS list for the current round, it begins executing tasks, each of which is represented as a service instance in SI. As indicated above, tasks require a service for a specified number of rounds (*reqt* in the task's requirement), so a service instance persists until *reqt* has elapsed, at which point the service instance is removed. If a completed task has subtasks, each subtask is added to the TASKS list so that they can be allocated or executed on the next round. In our model, each agent *connects* to a set of other agents, which are the only agents with which it can communicate, giving an organisational structure. Then, if an agent wants to find a service to which to allocate tasks, it performs a depth first search across the organisational structure until an appropriate service is found. Clearly the links between agents determine how easily an agent can find another agent offering a required service.

2.4 System Metrics

While any system can be designed to achieve different goals, in this paper we assume the most obvious goal of executing tasks as quickly as possible. In this subsection, therefore, we complete the description of our model by introducing the ways in which we are able to measure the performance of our system, first through three main metrics: *load*, *throughput*, and *messages*, and subsequently through ways of measuring other relevant systems properties.

The *load* of an agent is specified as $(|\mathrm{SI}|/cap) \times 100$, where $|\mathrm{SI}|$ is the number of service instances. The result is a percentage, indicating the degree of usage from full capacity at 100% to complete idleness at 0%. An agent is overloaded when its load is 100% and there are tasks in its TASKS list for which it satisfies the requirements. An agent's *throughput* is the number of tasks it *finishes* executing each round. The system's throughput is the total number of tasks that finish being executed at each time step. This value is suboptimal if tasks are waiting to be executed by one overloaded agent, but there are other agents that can execute these tasks immediately instead. Finally, when trying to locate a service to allocate a task that an agent cannot perform itself, the agent sends query messages to others for the required service. Similarly, to actually allocate a task, another message is sent. If the organisational structure is well designed, then agents will be close to the required services, thus sending fewer messages.

In addition to these three main metrics for performance, we can consider others that may be useful in what follows. For example, an agent's *task arrival rate* is the number of tasks that it receives each round, its *executable task arrival rate* is the number of tasks that it is capable of executing it receives each round, and its *allocation task arrival rate* is the number of tasks that it cannot execute it receives each round, and so must allocate elsewhere. We consider how frequently each service *type* is used, as *service frequency*.

3 Adapting Organisational Structure

The model just described specifies how tasks may be allocated and executed in distributed systems exemplified by our eScience scenario. However, since the structure of the system constrains the way in which tasks are allocated, we may not have an efficient system, or not as efficient as may be possible. For example, load (tasks requiring allocation and execution) may be poorly distributed across the agents (and their services) as a result. In consequence, this section describes a set of techniques that allow a system to be reorganised in such a way as to improve its efficiency in line with the metrics described above. In what follows, to be clear and to restate the difference between structure and topology, we take structure to be simply a set of connections between agents, and topology to be structure with some imposed constraints, to give rise to particular organisation types, such as pipelines, hierarchies, and so on.

Importantly, we want techniques that adapt a system so that efficiency is improved while also preserving the topology the system is designed to exploit. These techniques might apply to pipelines, hierarchies or other topologies, each of which has particular characteristics. In this respect we seek to provide a general form of reorganisation, a template that can be instantiated for different topologies, while at the same time drawing them all together in a coherent whole. In this paper we therefore discuss aspects of the general tools, but focus in particular on the case of simple pipelines.

In what follows, each reorganisation technique is designed around a two-stage process, where the first stage is concerned with analysing the specific organisational structure and determining what links between agents should be added to or removed, while the second stage ensures that the adaptation (the addition or removal of links) preserves the particular topology. As for task allocation, we assume that when a change is proposed to the structure, the agents that are included in the change will comply, in the sense that they will agree to the new structure.

In this paper, due to space constraints, we illustrate the general concept by means of one of the simplest topologies, *pipelines*, but seek to apply this template for reorganisation to other structures, in the same way, proposing changes to the links between agents, while preserving the overarching topology.

3.1 Analysing Structure

The first stage in reorganisation thus requires an analysis of an organisational structure in order to populate a *change set*, *C*, in which each element $c = (a_1, a_2, action) \in C$ indicates a change to be made, where a_1 and a_2 are agents that are either connected, or have the potential to be connected, and *action* is an element in the set $\{create, remove\}$. Now, since an organisational structure is, in essence, a graph defined by the *agents* (or vertices) involved and the *connections* (or edges) between them, the initial analysis uses graph metrics to find potential problems in the organisational structure, as follows.

By convention, a vertex is denoted by *v* such that $v \in \text{V}$ where V is the set of all vertices, and an edge is denoted by e such that $e \in \text{E}$ where E is a set of all edges. The end points of an edge are always two vertices, so an edge *e* is a vertex pair $e = (v_1, v_2)$. We thus represent a graph as $\text{G} = (\text{V}, \text{E})$ [11]. Similarly, we denote a multiagent system $\text{MAS} = (\text{A}, \text{C})$, where A is the set of all agents and C is the set of all connections between the agents in A. In a graph $\text{G} = (\text{V}, \text{E})$, a path is a sequence of vertices $P = (v_1, v_2, \dots)$ such that each vertex is connected to the next, in order, and the same vertex does not appear twice. A path is a traversal of a graph.

Given this description, we can introduce some standard graph metrics that may be used to undertake our initial analysis. First, the *degree of connectivity*, or the *degree*, of a vertex v is the number of edges that v is a part of, and is denoted by $deg(v)$. Second, the length of a path is the number of edges that it traverses. The *shortest path* is the path with the least number of edges, and the length of the shortest path between v_1 and v_2 is the *distance* between v_1 and v_2, denoted by $d(v_1, v_2)$.

Now, one of the key properties of an agent is its ability to interact with others. In this context, Freeman reviews different concepts of *centrality* [4], which is accepted as playing a significant role in influence in social networks and on the efficiency of group behaviour. For example, in Figure 3, an intuitive assessment suggests that *m* and *p* are central. According to Freeman, this is for three reasons: first, *m* has direct contact with the largest proportion of the system; second, *m* is closest to all other nodes in the system; and third, *m* and *p* are in control of much information that may pass between nodes [4]. Different measures of centrality can be defined: *degree centrality*, *closeness centrality*, and *betweenness centrality*.

Degree Centrality states an agent's centrality based on its degree, compared to the degree of all other agents. The higher the value the more central the agent. The degree centrality of an agent *m* is $\text{C}_d(m) = deg(m)$.

Betweenness Centrality is concerned with the position of an agent in relation to others. An agent *m* is *between* a pair of agents *n* and *q* if it appears in the shortest path connecting *n* and *q*. Betweenness centrality is the number of agent pairs that *m* appears between: the higher the value, the more central. In Figure 3, agent *m* is between seven agent pairs, while agent *p* is between six pairs, so agent *m* is the most central. Calculating *m*'s betweenness centrality is more complex when there is more than one shortest path between two agents, because *m* will no longer be between the two agents all of the time. If there are two shortest paths between *n* and *q*, and *m* appears in only one, then there is 0.5 chance that a message between *n* and *q* will pass through *m*. More specifically, *m*'s betweenness centrality can be measured by the number of shortest paths in which it appears, $\sigma_{nq}(m)$, divided by the total number of shortest paths, σ_{nq}, as in Equation 1. Szczepanski et al. [13] introduce a novel approach to more efficiently calculate betweenness centrality, but since we are interested in the use of betweenness, not its complexity, we do not consider this further.

$$\mathrm{C_B}(m) = \sum_{m \neq n \neq q \in \mathrm{A}} \frac{\sigma_{nq}(m)}{\sigma_{nq}} \tag{1}$$

Closeness Centrality gives an agent *m*'s centrality based on how close it is to the rest of the system. We can use Dijkstra's algorithm to find the shortest spanning tree rooted at agent *m*, consisting of the shortest path from *m* to all other agents. Agent *m*'s closeness centrality is the sum of the length of all of these shortest paths as in Equation 2; the lower, the value the more central.

$$\mathrm{C_C}(m) = \sum_{n \in \mathrm{A} \setminus m} d(m, n) \tag{2}$$

3.2 Proposed Changes to Structure

The above metrics provide a means of understanding certain properties of our organisational structures so that we are able to consider the changes that should be made. In the previous discussion, we have considered metrics in general, and continue this general analysis in this section in which we consider two simple ways of modifying a structure to give greater efficiency.

First, we seek to reduce the *distance* between an agent and the service it uses most frequently. Here, for each agent *m*, a depth-first search is used, starting at that agent, and searching through the entire organisational structure to find the closest instance of the service *s* that *m* most frequently uses, but that *m* and its neighbours do not offer. If the closest agent *n* offering service *s* is more than a specified number of hops away from agent *m*, then an element is added to the change set recommending a direct connection between *m* and *n*: $C = C \cup \{(m, n, create)\}$.

Second, we seek to decrease the load on overloaded agents by decreasing their closeness centrality, and moving them away from the centre of the system. Here, each agent's load is measured. If at least one agent has a load of 100%, with tasks waiting in its TASKS list that it has the capability to execute, then the system contains at least one overloaded agent. In response, each agent's centrality value is calculated and overloaded agents are moved one step away from the centre of the system, as follows. An overloaded agent *m* finds the neighbour *n* with the highest centrality, and the neighbour *p* with the lowest centrality. It also finds q, a neighbour of p with the lowest centrality out of *p*'s neighbours. A new connection is then created between m and q, while the connection between m and n is removed. The resulting elements in the change set are: $C = C \cup \{(m, q, create), (m, n, remove)\}$.

3.3 Preserving Topology

To this point, the techniques introduced, and indeed the methodology for doing so, have been presented in the most general way. However, this last part of the process requires

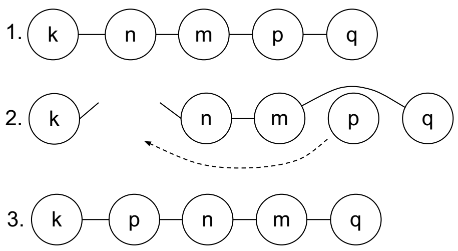


Fig. 4. An example of enacting a change in a pipeline

us now to instantiate our work with respect to a specific topology that constrains the changes proposed. In fact, as indicated in the introduction to the paper, the aim of our work is to facilitate reorganisation by building up a library of techniques and constraints in a stepwise fashion across different topologies, and then to generalise these instances to provide a generic model. This paper describes the early stages of this programme of work, and is restricted, as a first step, to *pipelines*, one of the simplest possible topologies, in order to illustrate the general approach. Subsequent work will consider application to hierarchies, matrices and other structures, but that is not considered in this paper.

In a pipeline, all of the vertices are lined up sequentially. Each vertex has a degree of connectivity of exactly 2, except for the vertices at the ends of the pipeline, which have exactly 1. In a multi-agent system, this translates to each agent having a maximum of two connections. If we wish to reorganise a pipeline, then we cannot change the number of connections, but we can change the connections themselves as long as we preserve the pipeline properties. The only legal change to such a structure is thus the positioning of each agent. Note that to do this we have only two possible changes from the change set: remove a connection and create (or add) a connection. However, if we remove a connection from a pipeline, we cannot preserve the structure since it breaks the pipeline irretrievably. For this reason, we do not entertain this possibility for pipelines (though we will do so in future work for other topologies), and focus here only on creating connections (though, confusingly, we will see that this will require removal of connections as part of the process of structure preservation).

To illustrate, suppose we have a pipeline with five agents, as shown in Figure 4 part 1, and a change set in which the first element states that a connection should be created between agents p and n. If this connection is created, then the organisational structure is no longer a pipeline. To ensure that the pipeline is maintained, the following changes must also be added to the change set: remove connections between m and p, q and p, and k and n, and create connections between m and q, and k and p. With all these changes, a connection can be created between p and n, while maintaining a pipeline as shown in Figure 4 part 3. In fact, this is one way to achieve the result we aim for, but the same can also be achieved by adding the connections elsewhere. For ease of exposition, we will not consider the alternatives in this paper, but note that they lead to the same outcome.

More formally, the change set is altered as follows: for each $(x, y, create) \in C$, x is moved next to y. As just noted, this is possible in different ways; here we do so by removing agent x, then reinsert x next to y in just one way.

Remove Agent. Removing x from a pipeline depends on x's position in the pipeline. If x has one connection then it is at the end of the pipeline, and so its only connection is removed. If x has two connections then it is in the middle of the pipeline, so both of its connections are removed. Then, a connection is created between x's previous neighbours, ensuring that the pipeline is maintained. In both instances, the result is a pipeline, excluding the single disconnected agent x.

Reinsert Agent. Reinserting x next to y similarly depends on y's position. If y has one connection then it is at the end of the pipeline, so a connection is created between x and y, and x is now at the end of the pipeline. If y has two connections then it is in the middle of the pipeline, so we must decide on which side of y to reinsert x. To do so we randomly select one of y's neighbours, z, remove the connection between y and z, and create connections between x and y, and x and z. In both instances, the result is a pipeline where x is directly connected to y.

As indicated above, we do not consider the removal of connections due to the constraints of a pipeline, since this cannot be done without irretrievably breaking the topology, and these changes are therefore simply eliminated from the change set.

Again, we assume that when a change is enacted, the agents involved in the creation or removal of the connection agree to the change. While it is of course possible to imagine scenarios in which this does not hold, for reasons of simplicity and clarity, we restrict our consideration only to such cases.

4 Evaluation

Given these techniques are designed to reorganise pipelines to increase the performance of a system, we undertook a series of experiments to show their impact and indeed whether they are successful. In order to provide a baseline for comparison, we experimented with *random* changes to the structure as well as the *reducing distance* and *decreasing load* changes.

As described above, changing a pipeline involves moving agents from one position to another. Whenever reorganisation is triggered in the random approach, 10 agents are randomly selected and moved next to another randomly selected agent that it is not currently connected to. Moving an agent a to an agent b consists of: removing the connection between a and each of its neighbours; creating a connection between each of a's old neighbours; removing the connection between b and one of its neighbours; and creating connections between a and b, and between a and b's old neighbour.

In what follows, we describe our results from simulating the task allocation and execution model described above. The simulation consisted of 100 agents in a pipeline with a random initial structure, where the agents receive tasks, and allocate or execute them. The simulation consisted of 2000 rounds, in each of which a number of tasks are randomly generated, and allocated to agents at random, regardless of the services offered by agents. The number of tasks generated at each time step varies according to the Poisson distribution with a mean task arrival rate of 10. On each round, agents follow the behaviour described in Section 2.3, after which reorganisation is potentially triggered: in rounds 0–499, there is no reorganisation, but in rounds 500–2000 reorganisation is triggered each time. Each simulation was repeated 15 times, with all results being averaged over these multiple runs.

4.1 Task Allocation and Execution

Throughout the simulation, the number of tasks that each agent is allocated and executes is counted, and the mean number across all agents plotted over time. It is important to note that a task is only considered to be *allocated* if it is delegated by another agent. Now, since tasks initially arrive at agents from outside of the system as a whole, and since some of these may be executed directly by those initial agents rather than being delegated onwards, this notion of allocation gives rise to a subtlety such that the number of tasks executed is very likely to be higher than the number of tasks allocated.

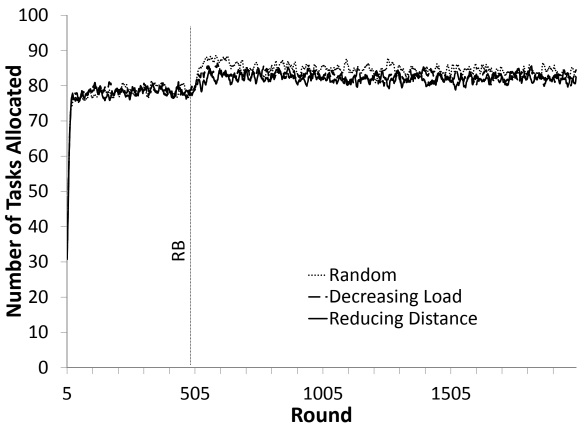


Fig. 5. Mean number of tasks allocated

Figure 5 shows how many tasks are allocated on average at each time step, and moreover that there is little variation between all three techniques, which each display a small increase in the number of tasks allocated. Random reorganisation shows a slightly larger increase, but the difference is minimal. Figure 6 shows how many tasks are executed on average at each time step. Again, there is little variation between the *reducing load* and *decreasing distance* techniques. However, we see better performance from random reorganisation.

In our model, locating services and sending a task between agents takes no time,[2] so the number of tasks executed increases because, rather than waiting on an overloaded agent's TASKS list, after reorganisation individual tasks are allocated to agents with spare capacity. In turn, this increases the number of subtasks being released for execution, so the number of tasks allocated also increases. The increase appears in all

[2] Instead, the number of messages needed, and the distance that tasks travel, are considered separately.

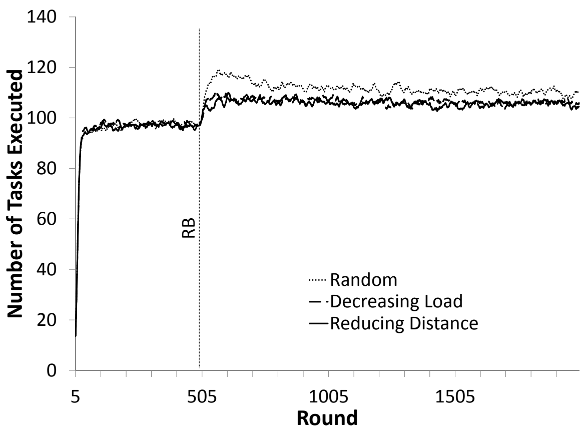


Fig. 6. Mean number of tasks executed

simulations, including *random*, indicating that the increase in task execution is due to the organisational structure continuously changing, rather than any specific change instance. Since an agent uses the relevant service on the closest agent, it will always use the same agent for a particular service unless the structure changes. Globally, this means that without any change to structure, agents whose services are not initially used will never be used, while those services that are initially used will be used regularly. However, by making regular changes to the organisational structure (random, or otherwise), the closest instance of a service to an agent will also regularly change.

4.2 Load and Waiting Tasks

Throughout the simulation, the load of each agent, the number of tasks waiting to be executed, across all agents, are recorded. Figure 7 shows the average load plotted over time, indicating that though all techniques increase the load of the system, random reorganisation improves loading the most significantly. Figure 8 shows the number of tasks waiting to be executed, with *reducing distance* and *decreasing load* both decreasing the rate at which tasks accumulate, but increasing the number of waiting tasks. In contrast, random reorganisation executes tasks faster than they arrive, so the number of tasks waiting to be executed decreases. However, this effect begins to plateau after 1,000 time steps. Overall, each technique offers some improvement, but random reorganisation is effective enough to address the accumulation of tasks before reorganisation. It is understandable that the first simulation does not significantly improve performance, since the desire to be closer to one instance of a required service does not aid in the distribution of load.

The second simulation tries to move overloaded agents away from areas of high centrality so that overloading can be avoided, and this seems to be only partially achieved. This could be for one of two reasons: either centrality is not a good enough indicator of the potential load of each agent, so using it to determine where to move an overloaded

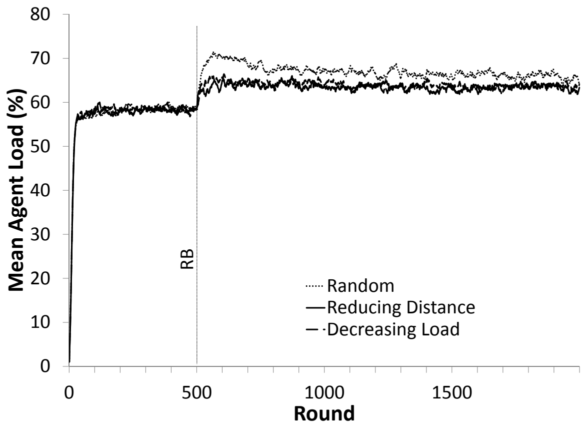


Fig. 7. Mean load of all agents

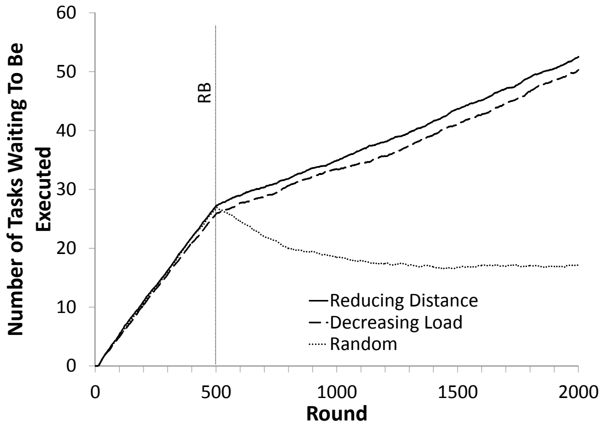


Fig. 8. Mean number of tasks waiting

agent is not sensible; or the reaction to finding an overloaded agent is not sufficient. In this latter case, instead of moving an overloaded agent away from areas of high centrality step by step, agents should be moved faster.

Random reorganisation performs best out of the three techniques, because the changes it makes are stronger, encouraging the use of multiple instances of services by changing the organisational structure, without changing the behaviour of agents. While we do not believe that random reorganisation is the most effective, it clearly has some properties that can improve the effectiveness of a reorganisation process.

4.3 Messages Sent and Average Distance

Our model does not directly account for the time spent locating services, and instead we consider this separately by counting the number of messages sent. Every message sent accounts for time that a task is waiting to be executed, either because a service is being located, or because the task is being moved to an agent that can execute it. Figure 9 shows the number of messages sent over time, indicating that this varies greatly from technique to technique. *Decreasing load* increases the number of messages required significantly, random reorganisation requires slightly more messages, and *reducing distance* successfully decreases the number of messages substantially. Figure 10 shows the reason for the difference in the number of messages, in terms of the average distance tasks are moved at each time step. With *decreasing load*, the distance from required services is increased, random reorganisation has no effect on the distance from required services, and *reducing distance* successfully moves agents closer to the services they require more frequently.

Though we previously showed that random reorganisation can execute tasks faster because it most effectively distributes load, this does not take into consideration the delay caused by locating services. Here, we can see that random reorganisation has no effect on the time required to allocate tasks, whereas by actively moving agents closer to the service they use most often, tasks can be allocated more quickly, reducing the time between a task being initially provided to the system, and the time the tasks start to be executed.

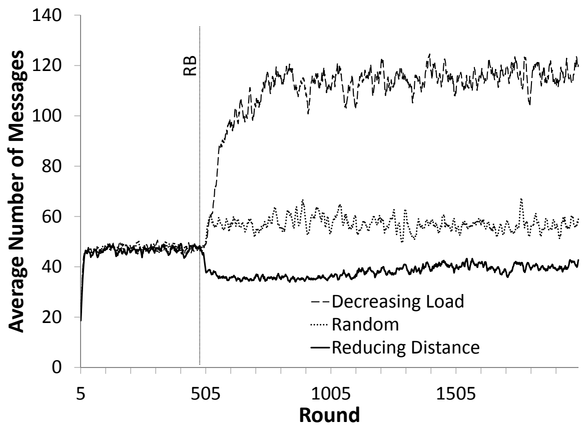


Fig. 9. The mean number of messages sent

4.4 Number of Changes

Figure 11 shows the number of connections changed in each time step while reorganising. Before round 500, no connections are changed, but afterwards, random reorganisation makes the least number of changes, which do not vary since this is fixed and

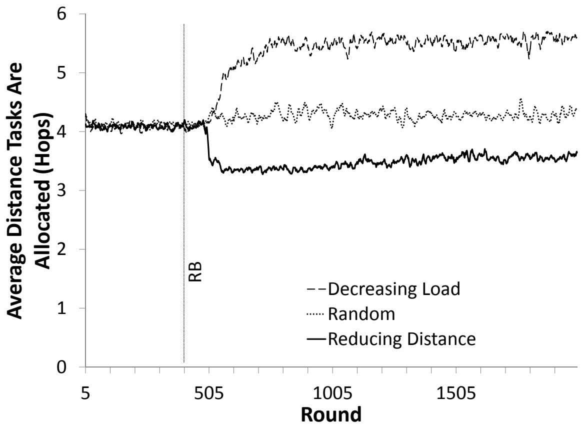


Fig. 10. The mean distance tasks are allocated

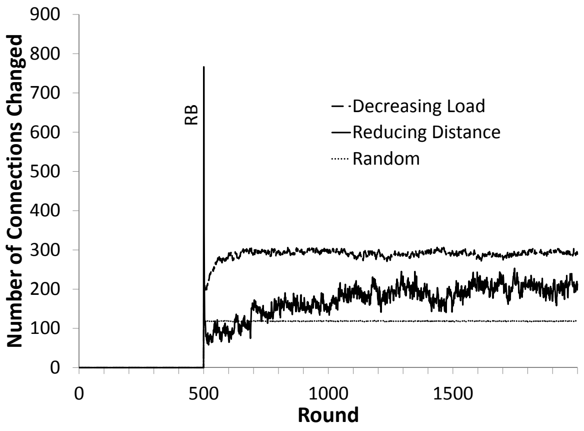


Fig. 11. The number of connections changed

not triggered by some conditions. *Reducing distance* initially makes a massive number of changes, changing nearly 800 connections, but this almost instantly falls to 100 connections each round, and then slowly rises to a plateau. The initial spike is due to the initial structure being badly organised, but this spike quickly falls when a better structure is found. However, reorganisation does not stop completely. This is because an agent *a* may have a service that many other agents desire, but a single agent can only have two neighbours at most, creating competition to be directly connected to agent *a*. In addition, every time an agent *a* moves, it potentially triggers its neighbours to move if *a*'s neighbours relied on a service *a* offered. *Decreasing load* has no initial spike, but instead rises to a plateau, making more changes than *reducing distance*.

4.5 Summary

We can see that the performance of each technique varies according to what we want to achieve. Random reorganisation is an effective way to encourage the use of multiple service instances, which in turn distributes load. However, this does nothing to increase the time between a task initially arriving, and the time at which execution begins. Nevertheless, each of our techniques can have a positive effect on different phases of the task allocation and execution lifecycle. Potentially by combining these techniques, and determining when to use each, we may be able to optimise reorganisation more effectively. Clearly the challenge lies in how to combine them and how to recognise when each is appropriate.

5 Related Work

Gershenson introduces reorganisation as a means of increasing the number of tasks a distributed system can execute, by decreasing communication delay arising from both transmission (latency of sending messages) and work to be performed before a reply can be sent (decision delay) [6]. In essence, this is concerned with locating the agent a that suffers the most from transmission and decision delays combined, the agent b that is a neighbour of a and causes the most decision delays, and the agent c that causes the least decision delays in the whole system. Then, the connection from a to b is removed, and a connection created from a to c. This technique was tested on a number of topologies (random-homogeneous, random-normally distributed, symmetrical, and scale-free), with results suggesting that: delay can be diminished, increasing the number of tasks executed; and the more connections, the longer to reorganise. While tackling a similar problem to this paper, Gershenson's consideration of topologies is not in their preservation but only their initial state.

Sims et al. introduce a self-organisation technique for a distributed sensor network that tracks the position and movement of vehicles [12]. Each sector involves a group of agents responsible for vehicles within the sector; problems arise when a vehicle moves along a boundary between sectors, requiring inter-sector communication with large overheads. Reorganisation here aims to minimise this communication by adjusting sector membership: if sector s_1 regularly needs information from a sensor in another sector s_2, and s_2 rarely uses it, then the sensor can be moved from s_1 to s_2, increasing global utility. This technique can adapt the organisational structure of a distributed sensor network, while maintaining the hierarchical topology in each sector. However, this is a basic two-tier hierarchy, and the changes do not consider the agent's position in the hierarchy, but rather the utility of an agent's capability (what area an agent can monitor). In contrast, our work is concerned with the organisational structure itself. Abdallah and Lesser [1,2], and later Zhang et al. [15,16], extend the work of Sims et al. and improve the self-organisation with reinforcement learning. While this improves the effectiveness of reorganisation, it still focuses on the performance of individual agents in the system, rather than on the organisational structure itself.

Gaston and desJardins introduce a couple of techniques to adapt the organisational structure of a set of agents so that teams can more easily be formed to execute tasks [5]. This is relevant to our work in providing an initial attempt at adaptation based purely on organisational structure, rather than on application-specific information. The results show that structural adaptation can be effective, and lead to better performance, but with a very high number of changes.

Similarly, Kota et al. introduce a reorganisation technique for the adaptation of problem-solving agent organisations [10,9] in which agents receive tasks that require services to be executed by an agent itself or by delegating to another agent. Here, reorganisation involves evaluating individual connections between agents based on performance or potential performance, and creating or removing connections appropriately. The performance of a connection is based on its effect on the load of the agents on either side, the change in the number of messages sent, and cost of reorganising, so that, for example, middlemen in lines of communication are removed. However, the structure of the organisation is never directly analysed, and again the process is based on the system which is application specific.

6 Conclusions

This paper has presented a novel method of reorganisation to optimise system performance, while at the same time preserving a topology. While the focus in the paper has been on the case of simple pipelines, the implications of the work, and the more general programme of research in which it is situated, are much more far-reaching. In terms of the specific techniques presented, it is interesting to note that random reorganisation turns out to be most successful at distributing load because it brings about radical rather than incremental changes: this is another case of the simple out performing the sophisticated in certain conditions. However, in the case of reducing messages, and decreasing the distance between agents and required services, random reorganisation has no effect, while reducing distance reorganisation shows a significant improvement on both counts.

The broad conclusions that we come to indicate that our techniques are sound in particular aspects as elaborated though the paper, as well as providing a template for further work on other topologies. Indeed, as part of a more general programme, this is just the first step. Pipelines are clearly one of the more simple topologies that can be analysed, yet this first effort shows the way forward in seeking to develop a general methodology for reorganisation as well as a library of techniques for particular purposes.

References

1. Abdallah, S., Lesser, V.: Learning The Task Allocation Game. In: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 850–857 (2006)
2. Abdallah, S., Lesser, V.: Multiagent reinforcement learning and self-organization in a network of agents. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1–8 (2007)
3. Argente, E., Julian, V., Botti, V.: Multi-Agent System Development Based on Organizations. Electronic Notes in Theoretical Computer Science 150(3), 55–71 (2006)
4. Freeman, L.: Centrality in Social Networks: conceptual clarification. Social Networks 1, 215–239 (1978/1979)

5. Gaston, M., des Jardins, M.: Agent-Organized Networks for Dynamic Team Formation. In: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 230–237 (2005)
6. Gershenson, C., Apostel, C.: Towards Self-organizing Bureaucracies. International Journal of Public Information Systems 2008(1), 1–24 (2008)
7. Hey, T.: The UK e-Science Core Programme and the Grid. Future Generation Computer Systems 18(8), 1017–1031 (2002)
8. Hey, T., Trefethen, A.: Cyberinfrastructure for e-Science. Science 308, 817–821 (2005)
9. Kota, R., Gibbins, N., Jennings, N.: Self-Organising Agent Organisations. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 797–804 (2009)
10. Kota, R., Gibbins, N., Jennings, N.: Decentralised Approaches for Self-Adaptation in Agent Organisations. ACM Transactions on Autonomous and Adaptive Systems 7(1) (2012)
11. Lipschutz, S.: Essential Computer Mathematics. McGraw-Hill Book Company (1982)
12. Sims, M., Goldman, C., Lesser, V.: Self-Organization through Bottom-up Coalition Formation. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 867–874 (2003)
13. Szczepanski, P., Michalak, T., Rahwan, T.: A New Approach to Betweenness Centrality Based on the Shapley Value. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, pp. 239–246 (2012)
14. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: the Gaia Methodology. ACM Transactions on Software Engineering and Methodology 12(3), 317–370 (2003)
15. Zhang, C., Abdallah, S., Lesser, V.: Integrating Organizational Control into Multi-Agent Learning. In: Proceedings of the 8th International Conference on Autonomous Agents and MultiAgent Systems, pp. 757–764 (2009)
16. Zhang, C., Abdallah, S., Lesser, V.: Self-Organization for Coordinating Decentralized Reinforcement Learning. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 739–746 (2010)

Author Index