Monitoring Norms

A Multi-Disciplinary Perspective

Dastani, Mehdi; Torroni, Paolo; Yorke-Smith, Neil

DOI
10.1017/S0269888918000267

Publication date
2018

Document Version
Final published version

Published in
The Knowledge Engineering Review

Citation (APA)
Dastani, M., Torroni, P., & Yorke-Smith, N. (2018). Monitoring Norms: A Multi-Disciplinary Perspective. *The Knowledge Engineering Review*, *33*, 1-22. Article e25. https://doi.org/10.1017/S0269888918000267

The Knowledge Engineering Review, Vol. 33, e25, 1–22.
doi:10.1017/S0269888918000267

Monitoring norms: a multi-disciplinary perspective

MEHDI DASTANI[1], PAOLO TORRONI[2] and NEIL YORKE-SMITH[3,4]

[1]*University of Utrecht, The Netherlands;*
e-mail: M.M.Dastani@uu.nl
[2]*DISI, University of Bologna, Italy;*
e-mail: paolo.torroni@unibo.it
[3]*Delft University of Technology, The Netherlands;*
e-mail: n.yorke-smith@tudelft.nl
[4]*American University of Beirut, Lebanon;*

Abstract

The concept of a *norm* is found widely across fields including artificial intelligence, biology, computer security, cultural studies, economics, law, organizational behaviour and psychology. The concept is studied with different terminology and perspectives, including individual, social, legal and philosophical. If a norm is an expected behaviour in a social setting, then this article considers how it can be determined whether an individual is adhering to this expected behaviour. We call this process *monitoring*, and again it is a concept known with different terminology in different fields. Monitoring of norms is foundational for processes of accountability, enforcement, regulation and sanctioning. Starting with a broad focus and narrowing to the multi-agent systems literature, this survey addresses four key questions: what is monitoring, what is monitored, who does the monitoring and how the monitoring is accomplished.

1 Introduction

If a *norm* is an expected behaviour in a social setting (Gibbs, 1965; Habermas, 1985; Bicchieri, 2006), then this article surveys how one can determine whether an individual is adhering to this expected behaviour—a process called *norm monitoring*.

The concept of a norm arises widely across diverse fields (Bicchieri & Muldoon, 2014) including artificial intelligence (AI) and computer science (e.g. security), cultural studies and sociology, economics and game theory, law and philosophy and organizational behaviour and psychology; and likewise the concept of norm monitoring arises widely. Hence this article seeks to adopt a common terminology, ask a set of common questions (below) and, in some case, draw out commonalities and differences across different fields and approaches.

Our foundation is the field of *multi-agent systems* (MASs), where norms hold a central role (Jennings, 1993) and norm monitoring is a 'central operational component' (Noriega *et al*., 2013). An MAS comprises distributed, autonomous reasoning entities called *agents*. For example, we can characterize the firms in a supply chain as agents. MAS is usually considered to be a subfield of AI, and can be seen also from a software engineering viewpoint; both in turn are subfields of computer science.

In MAS, norms are often considered as standards of behaviour used to control the overall system-level (i.e. social) behaviour. The term *normative MAS* is coined to refer to MASs where the overall system behaviour is controlled and coordinated by means of norms (Boella *et al*., 2006; Hollander & Wu, 2011). The enforcement of norms requires monitoring the agents' behaviour in order to detect and cope with norm violations. As we will review, some existing work on norm monitoring assumes a central entity that

observes the system behaviour, while other work proposes a (distributed) set of collaborating monitors that communicate and share their observations. Likewise, some monitoring approaches focus on observing global system states, while others focus on the performed actions or temporal system behaviour. Further, as we also survey in this article, various proposals make different assumptions about monitors. For example, some approaches assume imperfect or erroneous monitors while others assume perfect monitors with limited observation capability.

From a computational viewpoint, norm monitoring is closely related to existing work in control theory, where observability and controllability play a central role. Norm monitoring is also closely related to run-time verification, where system runs are examined against system requirements. Finally, norm monitoring is related to coordination approaches, where the execution of system components should be observed and coordinated, for example, by means of synchronization and coordination artefacts. Further, norms are studied in the social sciences such as in organizational behaviour, wherein the act of monitoring can be seen in a positive or negative light, and in fields such as law and philosophy, wherein norm violation is highly salient.

In the context of normative MAS, Alechina *et al*. (2013) remark: 'If our computational models are to advance, it might be time to start classifying the various models and tools. Which tools are best for designing/implementing online monitoring?' Recognizing this stated need, this article provides a selected review of models and tools, and addresses four key questions concerning monitoring of norms:

- what is monitoring and why monitor?
- what is monitored?
- who monitors?
- how to monitor?

We do not intend for a survey of norms or of normative MASs in general: see for instance Hollander and Wu (2011); Cranefield and Winikoff (2011); Andrighetto *et al*. (2013); Mahmoud *et al*. (2014). We present the perspective of the authors: we do not claim that we provide a unifying view, but we do provide an original perspective which attempts to consider norm monitoring with a wide angle.

We structure the article as follows. Section 2 surveys paradigms of monitoring in different fields. Section 3 goes deeper into operational models for monitoring in computer science. Section 4 identifies and collates challenges in norm monitoring.

2 Paradigms for norm monitoring

As we have already noted, the concept of 'norm' is found across sub-disciplines within the social and natural sciences. We begin with the former, but largely restrict our attention of the latter to the field of computer science.

2.1 Definitions and typologies of norms in the social sciences

Norms are commonly conceived as general standards of behaviour. In the literature, various norm types have been distinguished based on their attributes, purpose, uses and contexts. For instance, one classification key can be the 'authorities' that issue and enforce the norms (Ostrom, 1990).

An influential typology of norms is Gibbs's (1965) identification of 19 types of norms. While aiming to be comprehensive and while including monitoring, Gibbs admits that at least three of his important classification keys—including 'collective evaluation'—are problematic. As Interis (2011) further points out, part of Gibbs's difficulty is that he neglects a clear definition of sanctioning. Interis (2011) himself offers a simple typology, distinguishing between *descriptive norms* ('what "is" done') and *injunctive norms* ('what "should" be done'). The former have no sanctions, whereas the latter can have externally imposed sanctions or self-imposed sanctions. Interis's emphasis is norms as 'prevalent' behaviour in a social context.

Typical examples of norm types from these and other studies, including (Elster, 2009), are legal, social, moral and rational norms. We adopt these four as archetypal types of norms for which monitoring and

Table 1 Types of norms and their features

Norm type	Origin of norm	Monitoring subject	Purpose of monitoring	Need enforce?	Sanctioning	Scope	Encoding
Legal	Design, legislation	Institutional artifacts, agents empowered by society	To attain social goals	Yes	Administrative, pecuniary	Public	Formal
Social	Culture, emerging feature	Peers, society	To enable coordination, community governance	Maybe	Social stigma, reputation	Public	No
Moral	Design, cognitive justification	Background mental process, epistemic vigilance tools	To minimize epistemic incoherence	Subjective	Happiness, self-esteem	Private	Implicit or explicit
Rational	Mental processes	Agent or groups of agents	Descriptive/ analytic	No	(Varies)	Public/ private	Optional

related activities (e.g. sanctioning) differ. Table 1 summarizes them with an emphasis on monitoring. First, *legal norms* require a legislative body that issues norms (or provides for legal entities that can issue norms) and a corresponding executive body that enforces norms (Kelsen, 1991). For example, the legislative body of a state can issue a traffic law and the executive body of the state enforces the law. *Social norms* instead often emerge through interaction within a community of individuals, who are subsequently in charge of enforcing the norm. For example, the amount of labour in a workplace can emerge as a social norm, after which those who work too much or too little might be criticized or even ignored or excluded from the workplace. *Moral norms* differ from legal and social norms as there is no authority or society required to issue—explicitly or implicitly—and enforce moral norms. Rather, moral norms are seen as a product of reasoning or internalization of some external standards.[1] Individuals follow their moral norms because of internal reasons such as deliberation or emotions. Finally, *rational norms* include prescriptive rational rules such as axioms of logics or equilibria in games.

2.2 Monitoring in the social sciences

Social norms, being the customary rules that govern behaviour in groups and societies (Bicchieri & Muldoon, 2014), have been extensively studied in social science disciplines such as anthropology, economics, law, philosophy and sociology. Two distinct perspectives can be identified. One stream of research considers norms as *exogenous variables*, mainly seen as constraining behaviour—as with legal norms and conventions, central to the production of social order or social coordination. The focus here is on the function of norms and on the conditions under which norms will be obeyed. Under this perspective, work on norms in sociology and economics has produced theoretical and experimental studies into norm emergence, dynamics, convergence and evolution (Axelrod, 1986; Kameda *et al*., 2003; Ostrom, 2014). Philosophers have taken a different approach to norms, mainly considering them as *endogenous variables*: the unplanned, unexpected result of individuals' interactions. The focus in this case is on the emergence and dynamics of *conventions*, as solutions to recurring coordination problems (Lewis, 1969; Young, 1993), or *norms* in relation with beliefs, expectations, group knowledge and common knowledge (Bicchieri, 2006).

These two perspectives naturally lead to approaching aspects of norms such as conformance/compliance, sanctioning and enforcement in different ways. From an *economic* viewpoint, the ability to monitor and enforce agreements was recognized as one of the elements enabling the success of

[1] Values from religious belief can be a source of moral norms and, in the case of some religious systems, also a source of social norms (An-Na'im, 2011). The state can enshrine these norms to the status of legal norms and monitor them (Commins, 2015).

self-governance (Janssen & Ostrom, 2014). From a philosophical perspective, conformance to norms is best understood as the effect of individual awareness, preferences and expectations. In order for a social norm to exist, a sufficiently large group of individuals should both have *empirical expectations*, about others' conformance to the norm, and *normative expectations* about the others' expectations about one self's conformance to the norm. Here sanctioning could play a role in fostering normative expectations, thus monitoring becomes functional to the very existence of such norms. However, not all social norms involve sanctions, and sanctioning, retaliation or stigma are generally effective in small groups and in the context of repeated interactions, where the identity of the participants is known and monitoring behaviour is relatively easy (Bicchieri & Mercier, 2014; Bicchieri & Muldoon, 2014; Young, 2014).

We find different characterizations of norms and norm monitoring in a variety of other social science sub-disciplines. In *social psychology*, norms are attitudinal and behavioural uniformities among people, often considered in relation with the notion of conformity and the underlying group pressure, where key factors influencing the behaviour of social agents are the perception of the others' behaviour, the individual's attunement to the way one presents oneself in social situations, and the efforts to adjust one's performance to create the desired impression. Interestingly, in this context *self-monitoring* is the name given to the process of carefully controlling how one presents oneself. Thus monitoring is an inward, reflective process. Some broadly studied phenomena are conformity to (real or imagined) group pressure, compliance and obedience. Compliance, in particular, has a negative connotation, as it indicates a conformity that involves publicly acting in accord with social pressure, while privately disagreeing. Indeed, Hogg and Vaughan (2014) define compliance as a superficial, public and transitory change in behaviour and expressed attitudes in response to requests, coercion or group pressure.

In *political economy*, international affairs and developmental studies, where norms are mainly considered as exogenous variables, 'monitoring' of 'norms' takes meaning in the context of international treaties, for instance (Chapman, 1996): '"progressive realization," the current standard used to assess implementation [hence, "evaluating compliance with the norms"], renders economic, social, and cultural rights difficult to monitor. It is time for nongovernmental organizations, governments, and human rights monitoring bodies to reorient their work toward identifying and rectifying violations'. As such, the literature has little focus on formal models or questions of agency or psychology.

In another social science sub-discipline, *human resources* (Mathis & Jackson, 2010), monitoring can be defined as: to ensure that the performance objectives (key performance indicators) are being met in a timely and efficient manner. Here, monitoring has a positive and production-oriented connotation. An interesting connection with the computer science literature is Haynes *et al.* (2014), who study how the lack of a norm's support of goal can hinder organizational performance. By contrast, monitoring in the field of *organizational behaviour* (Griffin & Moorhead, 2014) can be defined as: to observe the behaviour of others (usually subordinates) to ensure compliance with formal or informal norms. Here, monitoring has a more negative or control/authority connotation (Frey, 1993; Hesselius *et al*., 2008). The same stance can be seen in economics and in principal-agent theory, as in Dye (1986).

It is worth observing that earlier works in different branches of the social sciences had great influence on the study of norms in the MASs literature, not only in domains born at the intersection of MAS and sociology, such as agent-based social simulation, but also in other areas of MAS research, such as coordination and electronic institutions, where, for example, seminal works in the field of economics (Ostrom, 1990) inspired later research on norm-governed MASs (Pitt *et al*., 2012).

2.3 Monitoring in software systems

The concept of monitoring permeates parts of computer science. We find it, for instance, in diverse sub-disciplines such as software engineering, databases, computer security, and event processing.

In the field of *software engineering*, monitors are mechanisms that aim at observing the execution or behaviour of software systems, in order to check whether they comply with a given set of requirements, norms, policies or constraints. Monitors are used in run-time verification and validation (V&V) of software systems because offline verification of software systems is often not feasible due to either the fact that their specifications/program codes are not available, or the fact that their offline V&V is computationally

intractable. Simple examples of such monitors are debuggers, tracers and profilers. Run-time V&V aims at monitoring an ongoing execution of a software system, in order to, for example, halt it whenever it deviates from a given set of desirable (safety and liveness) properties, requirements or norms. Thus monitors may not only observe, but also intervene. For example, monitors can intervene in an execution whenever the latter deviates from some desirable property: the monitor can then intervene by either halting the execution or inserting additional actions/instructions. A particular class of monitor is the *enforcement monitor*, which rewrites deviating executions to non-deviating executions.

In the field of *databases*, monitoring is often associated with the notion of *integrity constraints*. Database protocols have been devised in order to maintain integrity across multiple data sources. In particular, one important issue in *federated database systems* is checking integrity constraints over data from multiple sites in the federation. Integrity constraints specify those states of the (global) database that are considered to be semantically valid (Grefen & Widom, 1997). Active database rules, for instance, follow the event-condition-action (ECA) paradigm (Widom & Ceri, 1994) (*WHEN relevant update has occurred IF global constraint possibly violated THEN notify constraint manager*) and require an active rule manager to thus monitor and enforce the norms (constraints). Database *triggers* and *stored procedures* —which can be used in a similar way to ECA rules—are integrated functionalities in common database systems (e.g. Oracle, 2002).

In the field of *computer security*, and in particular in intrusion detection, several concepts are relevant to norm monitoring. In particular, one can distinguish between two complementary trends in intrusion detection: (1) the search for evidence of attacks based on knowledge accumulated from known attacks and (2) the search for deviations from a model of unusual behaviour based on observations of a system during a known normal state (Debar *et al*., 1999; Kemmerer & Vigna, 2002). The first trend is often referred to as *misuse detection* or *detection by appearance*. The second trend is referred to as *anomaly detection* or *detection by behaviour*. Anomaly detection uses models of the intended behaviour of users and applications, interpreting deviations from this 'normal' behaviour as a problem.

Monitoring is also a key concept in *complex event processing*—a generic computational paradigm in various application fields, ranging from data processing in web environments, to logistics and networking, to finance and medicine (Artikis *et al*., 2014). In this context, *event recognition* or *event pattern matching* (Luckham, 2002) refers to the detection of events that are considered relevant for processing, thereby providing the opportunity to implement reactive measures. Examples consist of the recognition of attacks in computer network nodes, human activities on video content, emerging stories and trends in online social networks, traffic and transport incidents in smart cities, fraud in electronic marketplaces, cardiac arrhythmias and epidemic spread. In each scenario, event recognition allows one to make sense of large data streams and react accordingly.

2.4 Monitoring in MASs

As we have seen, monitoring in normative MAS corresponds to monitoring the agents' behaviour in order to prevent, or detect and possibly sanction, norm violations. Supposing that the MAS is not *regimented*, that is, it is not possible to prevent all norm violations by agents, then the act of monitoring becomes fundamental. In other words, the need for monitoring in MAS derives from the autonomy of the agents. This issue has become increasingly salient in the last years with the progress made in branches of AI, coupled with the increasingly pervasive Internet-of-Things paradigm. Hence modern socio-technical systems are pushing in the direction of 'black-box' autonomy, as in robotic process automation and self-driving cars, which brings about the issue of accountability (Chopra & Singh, 2014; Gavanelli *et al*., 2018) and the calls for explainable AI. More generally, in open systems (Davidsson, 2001), where the regimented MAS model is not applicable, the meaning of agent communication is best described by what is externally observable (compare Section 3.1), giving rise to the so-called *social semantics* of agent interaction (Singh, 1998), which makes monitoring relevant not only in theory but also for a significant class of practical applications.

The interest in social norms is widespread in the MAS literature. Two significant early works are Shoham and Tennenholtz (1995) and the collection described by Conte *et al*. (1999). Even before then,

when MAS used to be called 'distributed artificial intelligence', authors such as Jennings 1993 argued that commitments and conventions are the 'foundation of coordination in multi-agent systems'. It should be noted, however, that Jennings used the term convention to refer to general policies for governing the reconsideration of an agent's own commitments, something akin to personal norms. The term 'convention' appears frequently in the MAS literature, however not in the sense intended by Jennings 1993, but rather in the way it is commonly intended in the evolutionary studies of norms, for example, in economics, in relation to social norms. See for example the classification of social norms from a game-theoretical perspective proposed by (Villatoro *et al.*, 2010).

This interest in social norms includes experimental studies about norms, such as the early work by Verhagen (2001)—an interest found also in the psychology and economics literature (Seinen & Schram, 2006; Hoffman *et al.*, 2007). The MAS literature, however, emphasizes the role of computational experiments of norm emergence and evolution. For instance, Campenni *et al.* (2010) study by simulation norm convergence, taking the view that a cognitive model underlies norms—in contrast to other works which have a perspective of model-less social learning and imitation. More recently, Morales *et al.* (2015) focus on automated norm synthesis, while Lorini and Mühlenbernd (2018) present a game-theoretic model of guilt in relation to sensitivity to norms of fairness.

In the context of MAS, then, norms are often considered as standards of behaviour used to control the overall system-level (i.e social) behaviour. The enforcement of norms requires monitoring the agents' behaviour in order to detect and cope with norm violations (Noriega *et al.*, 2013).

In more detail, Balke, da Costa Pereira, *et al.* (2013), introducing a collection of papers about normative MAS, cluster the uses of norms in MAS into three 'definitions'. First, 'the so-called "social science" definition looks at norms that are actively or passively transmitted and have a social function and impact'. That is, norms as a construct with social function.

Second, Balke, da Costa Pereira, *et al.* (2013) describe 'the so-called "norm-change" definition' which 'supports the derivation of those guidelines that motivate which definition of normative multi-agent system is used'. That is, norms as used to define the MAS.

Third, 'the so-called "mechanism design" definition entails the guidelines recommending the use and role of norms as a mechanism in a game-theoretic setting and to clarify the role of norms in the multi-agent system'. That is, norms as used within the MAS. The role of monitoring is greatest in this 'dimension'.

Moreover, as we will detail below, in the MAS literature, proposals for monitoring use a wide variety of formal frameworks such as control rules (Panagiotidi *et al.*, 2013), counts-as rules (Dastani *et al.*, 2017), event calculus (EC) (Yolum & Singh, 2004; Torroni, Chesani, *et al.*, 2009; Kafaland Torroni, 2012; Chesani *et al.*, 2013), model checking over temporal logic (Cranefield & Li, 2009), ontologies and semantic web rules (Fornara & Colombetti, 2010), Petri Nets (Jiang *et al.*, 2013), production rule systems (Alvarez-Napagao *et al.*, 2011), among others.

2.5 Summary

Looking across these disciplines, we see the frequency of the concept of norm and of the notion of norm monitoring. Summarizing, we can define norm monitoring as: the process of determining whether an individual is adhering to a set of norms. (We note in Section 4 the case of monitoring of a group of individuals.) While the rationale for monitoring varies, its importance lies in being an essential precursor to the subsequent question of what is done if norms are not being adhered to.

3 Models for norm monitoring

Having seen the existence and importance of norm monitoring across a range of fields in the social sciences and computer science disciplines, we now focus on computer science more closely, with attention to operational models for automated monitoring of norms. That is, we focus on computational models for norm monitoring.

There are various reasons for monitoring the behaviour of natural (i.e. human) or artificial (e.g. software) systems, and developing automated monitoring of them. For example, automatic monitoring of the

behaviour of visitors in public places such as museums—a natural system—is valuable for promoting conformance to positive social norms.[2] Indeed, the very existence of norms aimed at protecting the artworks (*Exhibits are not to be touched*) and at ensuring a cordial museum experience to all the visitors (*Use of mobile phones is prohibited*) is subordinate to the belief that failure to comply with norms will result in a sanction. Such a belief would mostly be unwarranted if no monitoring tool were thought to be in place. Likewise, automatic monitoring of business processes in a company enable compliance with business rules and polices, and automatic monitoring of the execution of software systems can assist to validate it or prevent it from malfunctioning. In all these applications, monitoring is a mechanism that observes and evaluates the behaviour of a system.

Computational models for monitoring of norms concern computational decision procedures that evaluate the observed behaviour (or execution) of natural or artificial systems with respect to a given set of norms to determine whether the system violates or complies with the norms. In order for a monitor to fulfil its purpose with respect to a target system, it requires the ability to observe and evaluate the behaviour of the target system. The input of monitoring mechanisms is thus the observable behaviour of a target system, often represented as a stream or trace of events, together with a set of norms that are used to evaluate the observed behaviour. A monitor should then analyse said behaviour and decide whether it violates any of the norms.

In computer science there is a distinction between *a-priori/static verification*, which is done offline, at design time, and *a-posteriori/run-time verification*, which is done either at execution time (online) or after execution. In particular, for a software system and a set of norms, it is possible to check the system compliance with the norms if the formal specification of the software system is available. In such a case, one may use various static verification techniques such as model checking techniques to determine the system's compliance (Clarke *et al*., 1999; Schnoebelen, 2002; Baier & Katoen, 2007). However, there are cases where either the specification of the software system is not available, as in open artificial societies (Davidsson, 2001), or the static verification techniques are computationally not attractive. In such cases, run-time verification and monitoring approaches can be used to ensure that norm violations during (or after) system executions are detected and properly treated. Run-time monitoring does not require any knowledge about the system's internal architecture, as it concerns the continuous observation and evaluation of a system's external behaviour as the system runs (Fisher *et al*., 2007). For these reasons, run-time or a-posteriori verification are the only possible types of verification that can be applied when formal specifications of a system are not available, for example when we consider natural or open systems (Bragaglia *et al*., 2012).

Formal and computational models of monitoring have been extensively studied in the field of run-time verification of software (Schneider, 2000; Bauer *et al*., 2011; Falcone *et al*., 2011) and open systems such as business processes (Montali, Torroni, *et al*., 2010; Ly *et al*., 2015), web choreographies (Alberti, Chesani, *et al*., 2006; Chesani *et al*., 2008; Bragaglia *et al*., 2011) and MASs (Alberti, Gavanelli, Lamma, Chesani, *et al*., 2006; Alberti *et al*., 2008; Modgil *et al*., 2015). The problem of run-time verification is formulated as checking whether a system execution satisfies or violates a given property, which represents a desired behaviour (e.g. a safety property). Since such properties are related to norms in the sense that both are properties that system executions can satisfy or violate, techniques from run-time verification have been used to check norm violations at run-time. In this section we survey some existing run-time verification and monitoring techniques in computer science which have been adopted—or could prove useful—in the development of computational norm monitoring systems in MASs.

3.1 What is monitored?

As we have seen, the goal of monitoring is to analyze the observed behaviour of a natural or artificial system, in order to match it with a given set of properties. Such properties can define norms. As we have seen in Section 2.2, norms could be expressed at a social or personal level, and they can have a prescriptive

[2] On the debate about privacy, monitoring of society, and security, see Helbing and Pournaras (2015); Zuboff (2015); Warnier et al. (2015); Baum (2017); Anderson (2018); Schneider (2018).

or descriptive nature. Social/legal norms are descriptive/prescriptive norms that gain meaning at a social level, as opposed to moral/rational norms, which gain a meaning at the personal level. For this reason, before we delve into the methods that enable run-time monitoring of norms, we should be clear about what is being monitored, which in turn depends on the nature of the norm we are focusing on.

If we consider personal norms, such as rational norms, the 'behaviour' to be observed is internal to the agent. This intra-agent deliberation ultimately manifests through the agent's actions in the (social) environment—but the deliberation itself is not externally manifest. For personal-level norms, then, we should assume we have access, at least to some extent, to the (artificial) agent's architecture and internal state, otherwise monitoring would be meaningless. In some cases we might even be able to turn monitoring into active control or regimentation by suitably controlling the procedures that govern the agent's behaviour.

By contrast, monitoring social/legal norms instead adopts the opposite assumption, that is, we do not know the agent's internal state. One typical setting is that of open artificial societies (Davidsson, 2001), where the architecture of the agents that constitute the system may be unknown; thus no direct intervention is possible on the procedures that govern the agent behaviour, the behaviour of the individual cannot be determined a priori, and the only possible kind of verification is a 'black-box' run-time or a-posteriori monitoring, solely based on the externally observable behaviour (Alberti, Chesani, *et al*., 2004).

3.1.1 Monitoring personal-level norms

If what is monitored is the internal behaviour of an (artificial) agent, then the desired behaviour of a system can be represented as a set of finite or infinite sequences of states or state transitions (events), which are often called *system traces*. (Since the literature focuses on monitoring traces, that is, trace-based norms as we might call them, we mention monitoring over event-based norms in Section 4.) The desired system behaviour is often formally modelled using regular expressions, finite state machines (FSMs), temporal formulae—for example, linear temporal logic (LTL), computation logic tree (CTL), alternating-time temporal logic (ATL)—some fragments of these temporal logics (Bulling *et al*., 2013), or a computer program. These formalisms are often concerned with infinite traces or executions in the sense that an expression from these formalisms denotes a set of infinite traces. For example, a LTL formula denotes a set of infinite traces that satisfy the formula. However, variations of these formalisms are proposed to deal with finite traces. For example, a LTL formula can be evaluated based on three-valued semantics such that a LTL formula denotes a set on finite traces (Bauer *et al*., 2010).

3.1.2 Monitoring social-level norms

If what is monitored is the externally-observable behaviour of agents whose internals cannot be inspected, then in fields such as complex event processing, business processes, web services, artificial societies and open systems in general, the behaviour that is monitored is typically a time-stamped sequence of sensor data, for example, *time series*, such as a (possibly large) *stream*. In complex event processing, the aim is often to establish connections between short-term and long-term activities. For example, in a city transport management setting, the behaviour could include heterogeneous streams of data from public vehicles and sensors mounted on intersections. Such data could be information about position and delay of vehicles, congestion, in-vehicle temperature, noise and passenger density. In such an example, a monitor may aim at recognizing complex events concerning bus punctuality, traffic congestion, traffic flow and density trends (for traffic congestion forecasting), and source disagreement, that is, when buses and SCATS[3] sensors provide conflicting information on traffic congestion.

Public space surveillance is another example of an application area for complex event processing. In such an application, the behaviour consists of video streams originating from CCTV and the monitoring aim is to detect 'short-term activities', that is, activities taking place in a short period of time detected on individual video frames, such as someone entering or exiting the surveillance area, walking, running, moving abruptly, being active or inactive and 'long-term activities', such as a person leaving an object

[3] www.scats.com.au

unattended, two people meeting, moving together and fighting. Short-term activities can be viewed as *low-level events* (also called simple, derived events, SDE), while long-term activities can be viewed as *complex events* (Artikis *et al*., 2015). In complex event processing, this inference from low-level streams up to complex events and long-term activities is clearly interesting for norm monitoring, where monitoring could be based on low-level streams. Norms, which must be simple in order to be of any effect, could describe which behaviours are permitted or prohibited at a much higher level of abstraction.

In MASs most of the focus has been on the monitoring of compliance to protocols and norms—a much higher level of abstraction than complex event processing. Protocols can be specified by expressing the legal sequences of actions using Petri Nets or FSMs (Jiang *et al*., 2013). However, advocates of declarative approaches maintain that interactions defined as sequences of states are over-constrained, which reduces autonomy, heterogeneity and potential to exploit opportunities and handle exceptions (Yolum & Singh, 2002). Similar stances have been taken in cognate domains such as business processes and web services (Pesic & van der Aalst, 2006; van der Aalst & Pesic, 2006; Montali, Pesic, *et al*., 2010). Moreover, the definition of interaction protocols or norms by way logical formulas enables exploiting a number of verification tools (El-Menshawy *et al*., 2015). Formalisms used to define protocols and norms include temporal logics (Bulling *et al*., 2013), the EC (Kowalski & Sergot, 1985) the reactive EC (REC) (Chesani *et al*., 2010), the lightweight coordination calculus (LCC) (Robertson, 2004), frameworks supporting social semantics for agent communication such as the SCIFF language of social expectations (Alberti *et al*., 2008), commitment manipulation languages (Singh, 1998) such as the EC-based one proposed by Yolum and Singh (2002) and REC-based commitment manipulation language (Torroni, Chesani, *et al*., 2009), contract specification languages such as defeasible logic-based business contract language (Governatori & Pham, 2009) and, especially in the context of norms, deontic logic (von Wright, 1951).

Figure 1 summarizes monitoring along several dimensions. One is the level of abstraction of input data, which could range from raw data streams to higher level descriptions. Another dimension is the norm scope: personal or social levels. A third dimension is the type of data that is observed: typically, it may be events (as in system traces), but it could also be states of the world. Finally, a fourth orthogonal dimension is the language in which the expected behaviour is stated: here the figure shows some of the possibilities.

3.2 Who monitors?

Norm monitoring can be done in various ways depending on the type of norms to be monitored, the characteristics of the system, and the (managerial) objectives of monitoring. As we saw in Table 1, the literature distinguishes various types of norms based on who issues and monitors norms (Bicchieri, 2006; Elster, 2009). For example, legal norms are construed as being issued by a legislative body and monitored

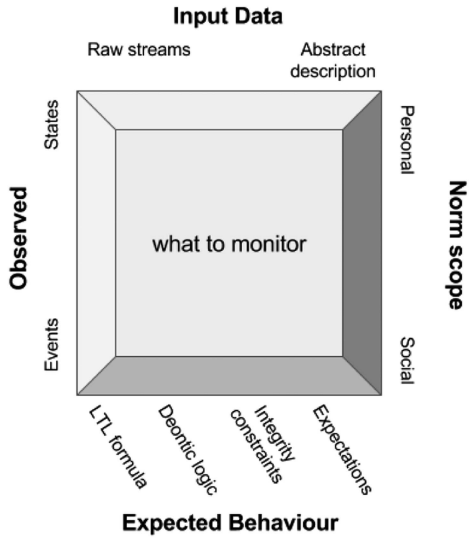

Figure 1 Dimensions defining what to monitor

by a corresponding executive body (which might be the same as the legislative body), while social norms are thought to be emerged from agents' interactions and monitored by the members of the society. Certain forms of legal norms, namely contracts and licences, have a social element in that they can be issued by a non-legislative body (e.g. a company) and be monitored by members of society (e.g. the same company)—but find their enforcement in a legal framework (e.g. civil court) established by a legislative body.

In contrast to legal and social norms, moral norms are defined as being issued and monitored by individual agents. These types of norms have inspired the design and analysis of computational systems. In particular, legal norms have been the inspiration for the design and development of exogenous norm-based coordination mechanisms in the sense that norms or requirements, which are issued and formulated by system designers, are used to synthesize monitors, which will subsequently observe and evaluate system executions. Social norms are often used to analyse the dynamics of systems, in particular, how system executions convergence in, or reach, equilibria states. For example, social norms are used to study the effect of interaction in various network settings. Finally, both social and moral norms have been an inspiration for designing norm-aware software systems, that is, software systems that adopt their behaviour to comply with norms or requirements. In the case of social norms, norm-aware agents may adapt their behaviour to comply with the prevailing social norm (Noriega *et al*., 2013), whereas in the case of moral norms, the adaptation has an inward motivation.

Different types of norms require different monitoring approaches. The traces of a target system can be monitored externally by means of a (software) system, which, in the case of MAS, can be another (software) agent; or internally by a mechanism integrated in the target system itself. In the former case, the monitor is external to the target system and observes the observable behaviour of the system to decide whether the behaviour violates some input norms. This is the approach taken for instance by the e-contract monitoring system and the green open innovation platform (GOI) socio-technical system whose architectures are shown in Figures 2(a) and (b). Alternatively, each participating software system (agent) may be endowed with a capability to observe the behaviour of other software systems (considered as the target system) and to respond in case of norm violation. This is the case depicted in Figure 2(c) showing an approach to distributed monitoring and diagnosis of contract executions. One critical issue in the latter relates to possible conflicts between an agent's goals—the end goals it wishes to achieve—and its monitoring goal. Such conflicts can result in strategic monitoring, which is an issue that deserves attention (Alechina *et al*., 2016; Alechina *et al*., 2017).

So far we have discussed external monitoring of artificial agents. Berger and Hevenstone (2016) report a study of monitoring of social norms in human societal contexts. These authors replicate and extend a pioneering field study of norm enforcement based on monitoring of natural agents, for norms about urban littering. The study finds that norms are enforced in three different cultural contexts, even when there is an enforcement cost to the monitor/enforcer (so-called *altruistic punishment*). In the second case, when monitors are internal and integrated in the target system, the monitor collects execution information, which is possibly used for the further continuation of the execution of the target system. An additional complexity comes from the distributed nature of such systems, due to which individual monitoring agents may have to coordinate with one another in order to diagnose norm violations (Kafaland & Torroni, 2012, 2018). Finally, since monitoring typically has to deal with large amounts of data and it is eminently an online activity, it crucially has to take computational verifiability into account, as Vázquez-Salceda *et al*. (2004) points out in relation with norm enforcement (but their analysis is more general). In particular, a norm may be non-verifiable not only because it is defined in terms of conditions or actions that are not observable, but also because its verification requires checking conditions or actions that in principle could be machine-verified, but in fact they are computationally hard to verify.

Monitors, whether they are external or internal, can have full or partial observation capacity (Bulling *et al*., 2013). In particular, a monitor for which two different traces of a target system may be indistinguishable is called a *partial monitor*. Indistinguishability can be due to the fact that a monitor may not have access to all information concerning an ongoing behaviour of a target system, for example, it cannot observe the entire state of, or all the actions performed by, the target system. Either way, a partial monitor cannot distinguish between all states or state transitions of the target system. In such cases, it is crucial to analyse the relation between monitors and the norms that should be complied with. Therefore, given a

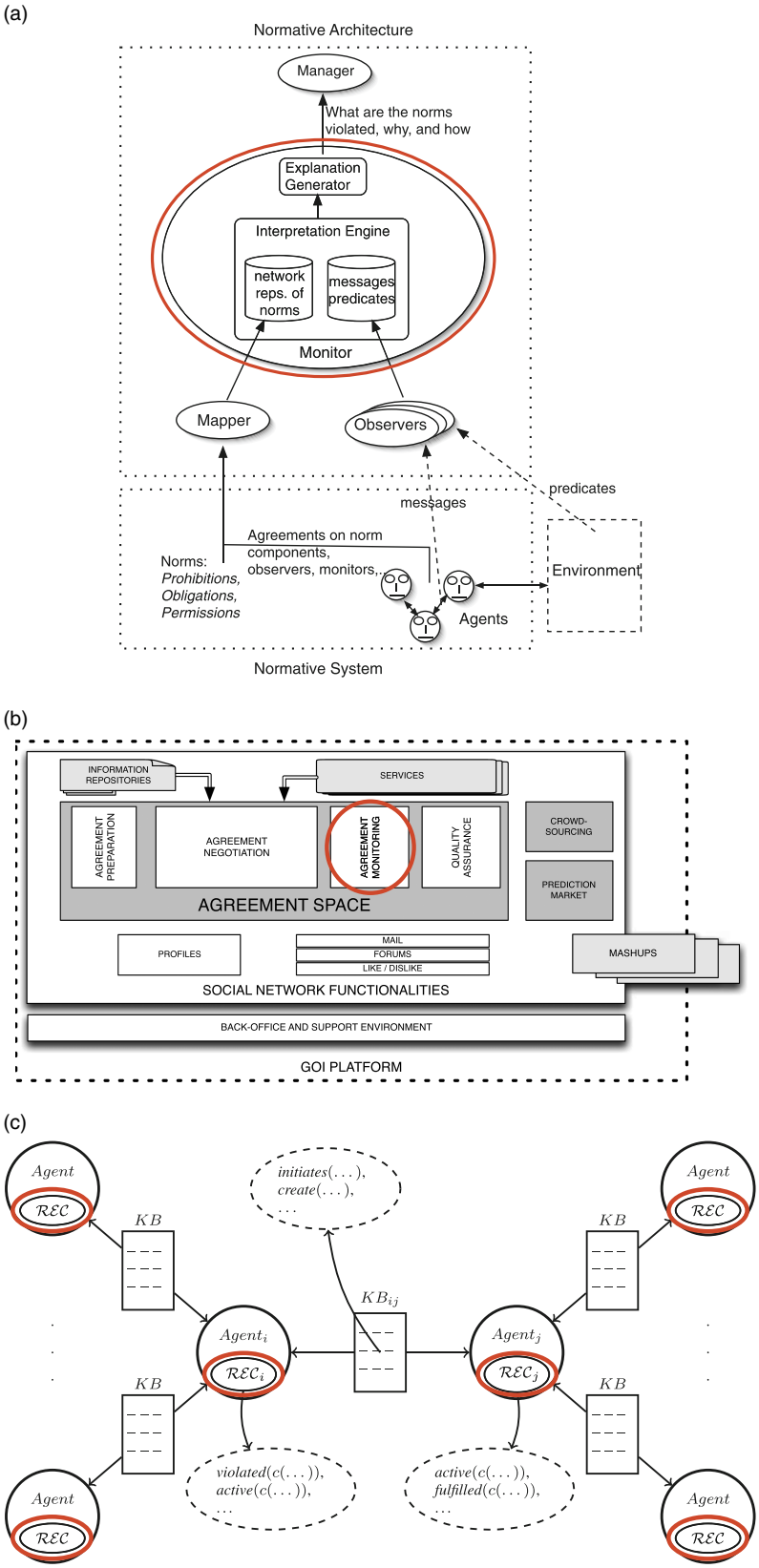

Figure 2 Example MAS architectures with norm monitoring. Monitors are highlighted with ovals. (a) e-contract monitoring (Modgil *et al.*, 2015); (b) Monitoring in the GOI sociotechnical system (Noriega *et al.*, 2013); (c) Distributed exception monitoring and diagnosis (Kafali and Torroni, 2013)

monitor and a set of norms, a natural question is whether the monitor can observe all norm deviations of a behaviour of a target system, and if not, which information are needed to perfectly observe all possible norm deviations. Such an analysis can help the system designer to extract more information from the behaviour of the target system. Alternatively, when additional information is not available or is expensive to obtain, the system designer may consider to weaken the given requirements such that the monitor can detect all possible norm deviations (Alechina *et al*., 2014).

Building on these ideas, Bulling *et al*. (2013) propose a formal framework to study and analyze monitors with respect to a given set of requirements. Different types of monitors (e.g. ideal, correct and faulty monitors) are proposed and their relations with the system requirements are analyzed. In their work, requirements are specified in LTL and monitors can be combined to construct complex monitors which can verify more complex properties. It is shown that the computational complexity of the problem to prove whether a monitor is sufficient for a given requirement (a violation is always correctly detected) is within the complexity bounds of LTL model checking.

3.3 How to monitor?

As we have seen, a monitor is generally conceived as a computational artefact that is either integrated in the target system and monitors its behaviour internally, or is exogenous to the target system and monitors its behaviour externally. For example, Kishon *et al.* (1991) propose to integrate run-time monitors in the standard semantics of programming languages such that the execution of a target program based on the derived monitoring semantics provides monitoring information without changing the target program behaviour. Another monitoring approach for run-time verification is to use monitors as exogenous artefacts (Thati & Rosu, 2005; Bauer *et al*., 2011) that observe and evaluate the ongoing execution trace of a target system. This is the approach followed by all the systems depicted in Figure 2 and, to the best of our knowledge, by most works in the MAS domain, since more emphasis is given to social-level norms than to personal-level norms. However, agent-based social simulation and MAS studies with strong social or behavioural science aspects can emphasize the latter (Balke, Cranefield, *et al*., 2013; Srour & Yorke-Smith, 2016).

This architectural aspect is not the only dimension along which monitor implementations differ from one another. Figure 3 illustrates four different dimensions. *Architecture* refers to the monitor architecture as just discussed, where a monitor could be internal (endogenous), or external (implemented as one or more exogenous artefacts), and in that case it can follow a centralized design, such as in Modgil *et al.* (2015); Noriega *et al*. (2013), or a distributed design, such as in Kafalı and Torroni (2012). Another dimension is the *observation* capability of the monitor, which could be limited, as it often is the case in

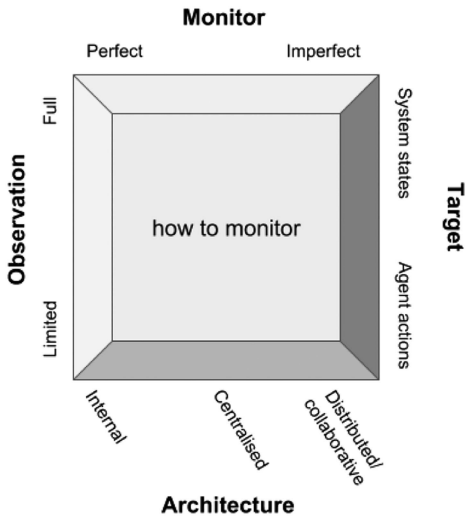

Figure 3 Dimensions defining how to monitor.

MASs due to the very nature of the domain, or full, as it happens when all the information the monitor needs is locally available. The *monitor* itself could be perfect or imperfect, as discussed previously, depending on the ability it has to distinguish between all/some of the system states. Finally, the *target* of monitoring indicates what the monitor is able or tuned to detect in its environment, which typically is system states or actions performed by agents. Here, the agent literature has mostly focussed on state-based norms (Alechina *et al*., 2013). This focus could be ascribed to action recognition posing harder challenges than state monitoring, which is basically a perception/sensing issue.

We emphasize here that the majority of normative MAS architectures consider monitors as an abstract facility, one which is seldom implemented. Great emphasis is given to the necessity of monitoring—not only in MAS but also in the larger literature we considered earlier—but not much is offered in terms of concrete formalisms or logics for monitoring and even less in terms of implementations. Influential MAS architectures and middlewares such as Islander (Esteva *et al*., 2002), MOISE + (Hübner *et al*., 2002), Ameli (Esteva *et al*., 2004) and others (Modgil *et al*., 2009; Noriega *et al*., 2013) often lack a concrete monitor implementation and often even a well-defined monitoring logic. In that sense, the architectures in Figure 2 can be seen as pioneering.

3.3.1 Internal monitors

To the best of our knowledge, little has been done with respect to monitoring personal-level norms of artificial systems. The problem however is very similar to that of run-time software verification, if we consider norms as properties that must never be violated by an agent's execution. Thus aspect-oriented programming (AOP) is a programming paradigm that can be exploited for norm monitoring purposes. AOP extends the object-oriented programming paradigm. The most mature AOP framework at the time of writing is AspectJ, which is specified for the Java programming language (Stolz & Bodden, 2006). AOP allows software developers to identify various points in program execution (*join points*), and to indicate the activities that should be performed when such execution points are reached. A join point can be the execution of a method or the assignment of a value to a variable. A selection of join points is specified by a *pointcut*. A pointcut might be activated during the program execution. An *advice* is a piece of code that specifies what needs to happen when a pointcut is activated. Thus AOP could be used to monitor the behaviour of a target system and to intervene, for example, to enforce some system properties or to collect execution information (Allan *et al*., 2005; Stolz & Bodden, 2006). Likewise, AOP could be used for norm monitoring and norm enforcement, if norms can be specified by means of pointcut regimentation, or if sanctioning can be modelled by combining pointcuts and advice.

3.3.2 External monitors

External monitors mean that monitoring of system execution is a process external to the system or agent being observed. A key source of inspiration for this type of monitors is *run-time verification* in computer science. In run-time verification, three stages are distinguished: monitor synthesis, system instrumentation and execution analysis. Given a correctness property, monitor synthesis generates a monitor, that is, here, a decision procedure, which can check the violation/compliance of the property against an execution of a target system. System instrumentation concerns how to feed relevant events of the target system into the monitor. Finally, execution analysis focuses on how an execution of the target system can be analyzed to decide whether the correctness property is satisfied or violated.

In run-time verification, a monitor for a correctness property is specified as a FSM. As a system evolves over time, observed events of the system are processed in the FSM, resulting in transition of the monitor's state. If the monitor reaches a state that evaluates the correctness property to be true/false/inconclusive, then the monitor is said to have detected respectively a compliance/violation/inconclusiveness of the norm. Such an approach can be found for instance in the proposed monitors for RV-LTL (Bauer *et al*., 2010) and in Westergaard (2011). The advantage of FSMs is that one can label states with additional information. For instance, Westergaard (2011) thus creates a system that allows the user to determine not only whether a norm is violated, but also exactly which transition caused this. However, a disadvantage

with the FSM approach is that the creation of the FSM is exponential in size given a set of LTL formulas and the amount of different possible events in the system.

In the related field of enforcement of security policies, various types of automata are used to monitor and enforce security policies (Schneider, 2000; Ligatti *et al*., 2005, 2009).

In the approaches described so far, the basic idea is that a monitor centrally observes the behaviour of a target system. In many applications, a central monitor may not be possible or may be available only at high expense. For example, the target system may be distributed such that a model of decentralized monitor is required. A central monitor creates a single point of failure, and it creates security risks because a central monitor may be the point where execution information are gathered for aggregation. Therefore, it is often desirable to monitor a target system (possibly a distributed target system) in a decentralized manner.

Monitoring a target system by *multiple monitors* has the advantage that the monitoring task can be distributed among different monitors in the sense that each monitor verifies a different set of norms. Alternatively, when the software system is distributed, one can deploy several monitors each of which verifies specific norms on various components of the target system. In both cases, monitors may need to interact to evaluate norms and to decide whether the software system deviates from some norms. A decentralized monitor consists of a set of collaborating monitors. In general, advantages of using decentralized monitors may include better scalability, graceful degradation in the face of system failures and parallel processing of sense data to increase the system's performance. Possible disadvantages include increased communication bandwidth, the introduction of a coordination overhead and need for maintaining data coherence.

One of decentralized monitoring approaches is based on the so-called *progression functions*. A progression function, as proposed in Bacchus and Kabanza (1998), has been an inspiration for some work on monitoring such as *hyMITL*$^{\pm}$ (Cranefield, 2006), and the works of Bauer and Falcone (2012) and Havelund and Rosu (2001). The core idea is, for a given state in a trace, to produce the formula which the next future state of the trace has to satisfy. The work in cranefieldspaper and havelund does not deal with decentralized monitoring. However the algorithm in decltlmon concerns decentralized LTL monitoring and assumes that all monitors can pairwise communicate and that all monitors are verifying one and the same global system requirement.

Other approaches to model decentralized LTL verification have been proposed by, for example, Bauer *et al*. (2011) and Testerink *et al*. (2015). While all monitors in Bauer *et al*. (2011) are connected to each other, Testerink *et al*. (2015) assume that monitors are connected according to a specific topology. In both proposals, the exchange of information takes time such that deviations from requirements are detected by monitors only with some delay. Another work on decentralized monitoring assumes each monitor verifies one specific set of requirements on one and the same system execution (Testerink *et al*., 2014). A characteristic feature of this work is its focus on the topology of the set of monitors. These monitors have to cooperate by exchanging information in order to detect the deviations from their requirements.

The MAS literature has also provided several alternative approaches to run-time verification. Notably, there exist logic-based tools that reason upon the externally observable behavior of interacting agents and verify whether it complies to predefined norms or protocols. In particular, the SOCS-SI tool (Alberti, Gavanelli, Lamma, Chesani, *et al*., 2006) uses the abductive logic programming SCIFF language and proof-procedure (Alberti *et al*., 2008) to consider messages exchanged by agents plus other events and carries out such a run-time interaction verification task. The SCIFF language is composed of entities for expressing *events*, *expectations* about events, and *relationships* between events and expectations. Events represent what is observable, and they could be elements of streams of execution traces. Expectations could be about events occurring (positive expectations) or about events not occurring (negative expectations). Events and expectations can be related together to form *social integrity constraints*, which specify what is expected given a certain execution trace. This is a powerful tool to model interaction protocols (Alberti, Daolio, *et al*., 2004) and norms (Alberti, Gavanelli, Lamma, Mello, *et al*., 2006).

Social commitments have become a popular tool for modelling social interactions, thereby providing a social semantics for agent communication languages, and defining multi-agent contracts (Castelfranchi, 1995; Singh, 1998; Singh, 1999; Fornara & Colombetti, 2003; Torroni, Yolum, *et al*., 2009; Fornara & Colombetti, 2010; Dastani *et al*., 2017). A social commitment is a commitment by a debtor towards a

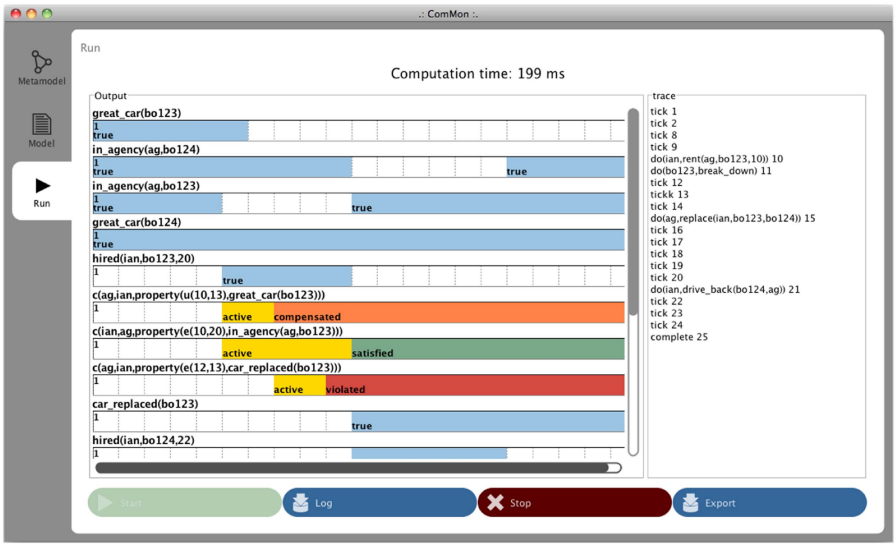

Figure 4 A screenshot of the ComMon run-time monitor for multi-agent commitments. Each row in the Output area shows a commitment over time; the coloured bars show the state of the commitment. The right-hand area shows the history of events, both environmental events (e.g. clock tick) and agent events (messages and actions).

creditor for bringing about a given property. Norms can be expressed in terms of commitments. Two existing tools that can be used for norm monitoring are jRec and ComMon. The former is based on Java and Prolog, and reasons upon the dynamics of an event-based system with respect to an EC specification. It acquires the event occurrences characterizing a running execution of the system and monitors their effects by updating the evolution of the corresponding fluents (i.e. properties varying over time) according to an EC theory.

As an example of a developed monitoring tool, we briefly describe ComMon, a run-time monitor for multi-agent commitments (Montali, Chesani, *et al.*, 2010), seen in Figure 4.[4] ComMon's commitment specification language is REC (Chesani *et al.*, 2010), an extension of the EC. ComMon enables the specification of properties that refer to a knowledge base, and reason with metric time, that is, 'real' deadlines (as opposed to time intended by temporal logic operators such as 'next' or 'until' and the like) (Chesani *et al.*, 2013). ComMon and related tools have been used in a number of works (e.g. Kafaland Torroni, 2012; Ly *et al.*, 2015; Kafaland Yolum, 2016; Kafaland Torroni, 2018).

Finally, since arguably the most widespread formalism for expressing legal norms is deontic logic (von Wright, 1951), in the MAS field there are several systems that implement deontic logic either to reason about norms, as was done by (Artikis *et al.*, 2009) in the context of the ALFEBIITE project, or to internalize norms, as it happens with IMPACT agents, where internal decision-making is governed by the application of permission and obligation operators (Subrahmanian *et al.*, 2000). There are fewer systems, however, that use deontic logic for monitoring norms at run-time. One such system was proposed by DBLP:journals/ail/ModgilOFMML15 who also produced a proof-of-concept implementation. Other systems such as SOCS-SI (Alberti, Gavanelli, Lamma, Chesani, *et al.*, 2006) use a logic programming framework with a deontic interpretation (Alberti, Gavanelli, Lamma, Mello, *et al.*, 2006). Outside of MAS, norm-related concepts such as access control policies in computer security are sometimes defined using semantic web languages like description logic or resource description framework, see Kirrane *et al.* (2017).

4 Open challenges in norm monitoring

Monitoring of norms has natural relevance to business, governmental and legal situations where agreements, contracts, law, protocols or regulations apply. For instance, norms have proved valuable as a

[4] From `https://www.inf.unibz.it/montali/tools.html#ComMon`.

flexible tool for Business Process Modelling and service engagement modelling, to establish compliance with, for instance, protocols. A challenge in monitoring such norm-based models is taking advances in norm monitoring from the academic literature and applying it to real-world, semi-cooperative circumstances (Giblin *et al*., 2006; Kalia *et al*., 2013; Singh *et al*., 2013), and further to convince senior decision makers of the value of norm-based modelling.

The business applications of norm monitoring and, as Alechina *et al*. (2013) point out, also the legal applications, need not only formal models—language and semantics—as surveyed in Section 3, but, since agent behaviours are 'usually shaped as narratives of events, i.e., traces', the need is for 'trace-friendly semantics'. This is a challenge for the theory and the practical application. Since different application domains have different monitoring constraints and requirements, Alechina *et al*. call for a theory underlying a family of languages, such that for each application domain the expressivity requirements can be met. The same authors also point out that current approaches to normative agent programming focus on state-based norms (and their monitoring), neglecting action- or event-based norms (and their monitoring).

When we surveyed operational models for monitoring in Section 3, several challenges were evident. These include imperfect or erroneous monitors, limited observational capability, probabilistic monitoring guarantees, strategic monitoring, and language expressiveness versus computational complexity of monitoring. Normative systems, in particular, need scalable architectures for behaviour identification and norm monitoring. In the words of Alechina *et al*. (2013):

> *One approach is to distribute such functions [particularly for distributed MAS], however this entails additional complexity, e.g., transactional approaches to state update. For very large scale systems, 100% monitoring may be infeasible, and probabilistic guarantees of the detection of norm violations may be more appropriate. Such approaches, in turn, may impact the design of sanctions, and, when applied to behaviour monitoring, the design of the conditional norms which specify the normative exogenous organisation itself.*

Another challenge relates to the motivational and even strategic aspect of monitoring in MASs: why should an agent cooperate in monitoring? First, monitoring has a cost to the agent(s). Second, there may be strategic reasons for an agent against providing accurate monitoring outputs. Principles of mechanism design and peer-to-peer architectures could represent a source of inspiration towards approaching the motivational challenge.

A further interesting venue for future research on norm monitoring is monitoring those norms having as creditor a group of agents, as opposed to an individual agent, especially when the regulated action is an action that required cooperation among the agents.[5]

Monitoring of norms reveals whether and how a violation occurs. There are challenges that follow should a violation be predicted or detected. These include blame assignment—since the agent that actually violates the norm may not be the guilty party—and, for non-regimented systems, sanctions, and reputation or incentives mechanisms and their design. These issues are growing increasingly salient as autonomous AI systems are becoming more pervasive, which calls for technologies that enable computational accountability (Baum, 2017; Chopra & Singh, 2014). For systems involving humans, a further challenge is the effect of culture (Dechesne *et al*., 2012).

5 Conclusion

This survey recognizes that monitoring of norms is foundational for processes of accountability, enforcement, regulation, and sanctioning in social settings, whether natural, artificial, or hybrid systems. Starting with a broad focus and narrowing to the MASs literature, we addressed four key questions: what is monitoring, what is monitored, who does the monitoring and how the monitoring is accomplished.

Monitoring and associated processes are an essential component of the fourth part of the generalized norm lifecycle (Savarimuthu & Cranefield, 2011; Andrighetto *et al*., 2013): creation, identification, spreading, enforcement, emergence and management of norms. We observed a wide range of techniques

[5] We thank an anonymous reviewer for this point.

used for formalizing a representation of norms in the MASs literature, and for their subsequent monitoring. We also contributed by identifying several dimensions in norm monitoring, which can help better position new research contributions.

We identified and collated critical open challenges in norm monitoring. In addition to these are general challenges are domain- or application-specific challenges, as discussed by Singh *et al*. (2013). Our survey suggests that those working on operational norm monitoring can benefit from the perspectives and the social science insights from, for example, sociology, while the mathematically rigorous MAS and computer science research can benefit the discourse in social science. Having noted the different understandings of norms—for example, the differences between legal and social norms—we encourage the MAS community to bear in mind the distinctions between norm types.

While there are some existing tools and methods (see e.g. jRec and ComMon discussed in Section 3.3.2) that are used for monitoring of norms and also for monitoring in a number of cognate domains, such as social commitments, business processes, interaction protocols, and web services, there is also a recognized need for a monitoring module in several social MAS architectures. However, we find that concrete implementations rarely invest in such a component. We hope to see this change future MAS infrastructures. Further, we find a manifest need for tracing/monitoring facilities in business infrastructures,[6] and we believe that investing in this aspect could help provide a new momentum to applied MAS technology.

Acknowledgements

We thank the reviewers for their suggestions which improved the article. Thanks to C. M. Karam. NYS was partially supported by grant number 102853 from the University Research Board, American University of Beirut. NYS thanks the fellowship at St Edmund's College, Cambridge.

References

Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Montali, M., Storari, S. & Torroni, P. 2006. Computational logic for run-time verification of web services choreographies: Exploiting the *SOCS-SI* tool. In *Proceedings of the 3rd International Workshop on Web Services and Formal Methods (WS-FM'06)*, *LNCS* 4184, 58–72. Springer.

Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P. & Torroni, P. 2004. The SOCS computational logic approach to the specification and verification of agent societies. In *Proceedings of the International Workshop on Global Computing (GC'04)*, *LNCS* **3267**, 314–339. Springer.

Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P. & Torroni, P. 2008. Verifiable agent interaction in abductive logic programming: the SCIFF framework. *ACM Transactions on Computational Logic* **9**, 29:1–29:43.

Alberti, M., Daolio, D., Torroni, P., Gavanelli, M., Lamma, E. & Mello, P. 2004. Specification and verification of agent interaction protocols in a logic-based system. In *Proceedings of the 19th ACM Symposium on Applied Computing (SAC'04)*, 72–78. ACM.

Alberti, M., Gavanelli, M., Lamma, E., Chesani, F., Mello, P. & Torroni, P. 2006. Compliance verification of agent interaction: a logic-based software tool. *Applied Artificial Intelligence* **20**, 133–157.

Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P. & Sartor, G. 2006. Mapping deontic operators to abductive expectations. *Computational & Mathematical Organization Theory* **12**, 205–225.

Alechina, N., Bassiliades, N., Dastani, M., Vos, M. D., Logan, B., Mera, S., Morris-Martin, A. & Schapachnik, F. 2013. Computational Models for Normative Multi-Agent Systems. In (Andrighetto et al., 2013), 71–92.

Alechina, N., Dastani, M. & Logan, B. 2014. Norm approximation for imperfect monitors. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'14)*, 117–124. IFAAMAS.

Alechina, N., Halpern, J. Y., Kash, I. A. & Logan, B. 2016. Decentralised norm monitoring in open multi-agent systems. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'16)*, 1399–1400. IFAAMAS.

Alechina, N., Halpern, J. Y., Kash, I. A. & Logan, B. 2017. Incentivising monitoring in open normative systems. In *Proceedings. of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, 305–311. AAAI.

Allan, C., Avgustinov, P., Christensen, A. S., Hendren, L., Kuzins, S., Lhoták, O., de Moor, O., Sereni, D., Sittampalam, G. & Tibble, J. 2005. Adding trace matching with free variables to AspectJ. In *Proceedings of the 20th*

[6] See for instance opentracing.io.

Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA'05), 345–364. ACM.

Alvarez-Napagao, S., Aldewereld, H., Vázquez-Salceda, J. & Dignum, F. 2011. Normative monitoring: semantics and implementation. In *Proceedings of the 6th International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN'10), LNCS* 6541, 321–336. Springer.

An-Na'im, A. A. 2011. Religious norms and family law: is it legal or normative pluralism? *Emory International Law Review* **25**, 785–810.

Anderson, R. 2018. Making security sustainable. *Communications of the ACM* **61**, 24–26.

Andrighetto, G., Governatori, G., Noriega, P. & van der Torre, L. W. N. eds. 2013. *Normative Multi-Agent Systems, Dagstuhl Follow-Ups 4*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. `http://drops.dagstuhl.de/opus/portals/dfu/index.php?semnr=13003`

Artikis, A., Gal, A., Kalogeraki, V. & Weidlich, M. 2014. Event recognition challenges and techniques: guest editors' introduction. *ACM Transactions on Internet Technologies* **14**, 1:1–1:9.

Artikis, A., Sergot, M. J. & Paliouras, G. 2015. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering* **27**, 895–908.

Artikis, A., Sergot, M. J. & Pitt, J. V. 2009. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic* **10**, 1:1–1:42.

Axelrod, R. 1986. An evolutionary approach to norms. *The American Political Science Review* **80**, 1095–1111.

Bacchus, F. & Kabanza, F. 1998. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence* **22**, 5–27.

Baier, C. & Katoen, J.-P. 2007. *Principles of Model Checking*. MIT Press.

Balke, T., Cranefield, S., di Tosto, G., Mahmoud, S., Paolucci, M., Savarimuthu, B. T. R. & Verhagen, H. 2013. Simulation and normas. In (Andrighetto et al., 2013), 171–189.

Balke, T., da Costa Pereira, C., Dignum, F., Lorini, E., Rotolo, A., Vasconcelos, W. & Villata, S. 2013. Norms in MAS: definitions and related concepts. In (Andrighetto et al., 2013), pages 1–31.

Bauer, A. & Falcone, Y. 2012. Decentralised LTL monitoring. In *Proceedings of the Formal Methods 2012 (FM'12)*, 85–100. Springer.

Bauer, A., Leucker, M. & Schallhart, C. 2010. Comparing LTL semantics for runtime verification. *Journal of Logic and Computation* **20**, 651–674.

Bauer, A., Leucker, M. & Schallhart, C. 2011. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engingeering Methodologies* **20**, 14:1–14:64.

Baum, S. D. 2017. On the promotion of safe and socially beneficial artificial intelligence. *AI & Society* **32**, 543–551.

Berger, J. & Hevenstone, D. 2016. Norm enforcement in the city revisited: An international field experiment of altruistic punishment, norm maintenance, and broken windows. *Rationality and Society* **28**, 299–319.

Bicchieri, C. 2006. *The Grammar of Society: The Nature and Dynamics of Social Norms*. Cambridge University Press.

Bicchieri, C. & Mercier, H. 2014. Norms and beliefs: how change occurs. In Xenitidou and Edmonds, (2014), 37–54.

Bicchieri, C. & Muldoon, R. 2014. Social norms. In *The Stanford Encyclopedia of Philosophy,* Zalta, E. N. (ed). `https://plato.stanford.edu/cgi-bin/encyclopedia/archinfo.cgi?entry=social-norms`.

Boella, G., Van Der Torre, L. & Verhagen, H. 2006. Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory* **12**, 71–79.

Bragaglia, S., Chesani, F., Fry, E., Mello, P., Montali, M. & Sottara, D. 2011. Event condition expectation (ECE-) rules for monitoring observable systems. In *Proceedings of the 5th International Conference on Rule-based Modeling and Computing on the Semantic Web (RuleML'11)*, *LNCS* 7018, 267–281. Springer.

Bragaglia, S., Chesani, F., Mello, P., Montali, M. & Torroni, P. 2012. Reactive event calculus for monitoring global computing applications. In *Logic Programs, Norms and Action*, *LNCS* 7360, 123–146. Springer.

Bulling, N., Dastani, M. & Knobbout, M. 2013. Monitoring norm violations in multi-agent systems. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, 491–498.

Campenni, M., Cecconi, F., Andrighetto, G. & Conte, R. 2010. Norm and social compliance: a computational study. *International Journal of Agent Technology Systems* **2**, 50–62.

Castelfranchi, C. 1995. Commitments: from individual intentions to groups and organizations. In *Proceedings of the 1st International Conference on Multiagent Systems (Agents'95)*, 41–48.

Chapman, A. R. 1996. A "violations approach" for monitoring the international covenant on economic, social and cultural rights. *Human Rights Quarterly* **18**, 23–66.

Chesani, F., Mello, P., Montali, M. & Torroni, P. 2008. Verification of choreographies during execution using the reactive event calculus. In *Proceedings of the 5th International Workshop on Web Services and Formal Methods (WS-FM'08)*, *LNCS* 5387, 55–72. Springer.

Chesani, F., Mello, P., Montali, M. & Torroni, P. 2010. A logic-based, reactive calculus of events. *Fundamenta Informaticae* **105**, 135–161.

Chesani, F., Mello, P., Montali, M. & Torroni, P. 2013. Representing and monitoring social commitments using the event calculus. *Autonomous Agents and Multi-Agent Systems* **27**, 85–130.

Chopra, A. K. & Singh, M. P. 2014. The thing itself speaks: Accountability as a foundation for requirements in sociotechnical systems. In *Proceedings. of the 7th International Workshop on Requirements Engineering and Law (RELAW'14)*, 22. IEEE.

Clarke, E., Grumberg, O. & Peled, D. 1999. *Model Checking*. MIT Press.

Commins, D. 2015. *Islam in Saudi Arabia*. Cornell University Press.

Conte, R., Falcone, R. & Sartor, G. 1999. Introduction: agents and norms: how to fill the gap? *Artifical Intelligence and Law* **7**, 1–15.

Cranefield, S. 2006. A rule language for modelling and monitoring social expectations in multi-agent systems. In *Proceedings of the 1st International Workshop on Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems (COIN'05)*, *LNCS* 3913, 246–258. Springer.

Cranefield, S. & Li, G. 2009. Monitoring social expectations in Second Life. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, 1303–1304.

Cranefield, S. & Winikoff, M. 2011. Verifying social expectations by model checking truncated paths. *Journal of Logic and Computation* **21**, 1217–1256.

Dastani, M., van der Torre, L. & Yorke-Smith, N. 2017. Commitments and interaction norms in organisations. *Autonomous Agents and Multi-Agent Systems* **31**, 207–249.

Davidsson, P. 2001. Categories of artificial societies. In *Proceedings of the 2nd International Workshop on Engineering Societies in the Agents World (ESAW'01)*, *LNCS* 2203, 1–9. Springer.

Debar, H., Dacier, M. & Wespi, A. 1999. Towards a taxonomy of intrusion-detection systems. *Computer Networks* **31**, 805–822.

Dechesne, F., Di Tosto, G., Dignum, V. & Dignum, F. 2012. No smoking here: values, norms and culture in multi-agent systems. *Artificial Intelligence and Law* **21**, 79–107.

Dye, R. A. 1986. Optimal monitoring policies in agencies. *The Rand Journal of Economics* **17**, 339–350.

El-Menshawy, M., Bentahar, J., Kholy, W. E., Yolum, P. & Dssouli, R. 2015. Computational logics and verification techniques of multi-agent commitments: survey. *Knowledge Engineering Review* **30**, 564–606.

Elster, J. 2009. Social norms and the explanation of behavior. In *The Oxford Handbook of Analytical Sociology*, Hedström, P. & Bearman, P. (eds), 195–217. Oxford University Press.

Esteva, M., de la Cruz, D. & Sierra, C. 2002. ISLANDER: an electronic institutions editor. In *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, 1045–1052.

Esteva, M., Rosell, B., Rodrguez-Aguilar, J. A. & Arcos, J. L. 2004. AMELI: an agent-based middleware for electronic institutions. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, 236–243. IFAAMAS.

Falcone, Y., Mounier, L., Fernandez, J.-C. & Richier, J.-L. 2011. Runtime enforcement monitors: composition, synthesis, and enforcement abilities. *Formal Methods in System Design* **38**, 223–262.

Fisher, M., Bordini, R. H., Hirsch, B. & Torroni, P. 2007. Computational logics and agents: a road map of current technologies and future trends. *Computational Intelligence* **23**, 61–91.

Fornara, N. & Colombetti, M. 2003. Defining interaction protocols using a commitment-based agent communication language. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, 520–527.

Fornara, N. & Colombetti, M. 2010. Representation and monitoring of commitments and norms using OWL. *AI Communications* **23**, 341–356.

Frey, B. S. 1993. Does monitoring increase work effort? The rivalry with trust and loyalty. *Economic Inquiry* **31**, 663–670.

Gavanelli, M., Alberti, M. & Lamma, E. 2018. Accountable protocols in abductive logic programming. *ACM Transactions on Internet Technology* **18**, 46:1–1:20.

Gibbs, J. P. 1965. Norms: The problem of definition and classification. *American Journal of Sociology* **70**, 586–594.

Giblin, C., Müller, S. & Pfitzmann, B. 2006. From regulatory policies to event monitoring rules: towards model-driven compliance automation. *IBM Research Zurich, Report RZ*, **3662**. IBM.

Governatori, G. & Pham, D. 2009. DR-CONTRACT: An architecture for e-contracts in defeasible logic. *International Journal of Business Process Integration and Management* **4**, 187–199.

Grefen, P. W. P. J. & Widom, J. 1997. Protocols for integrity constraint checking in federated databases. *Distributed and Parallel Databases* **5**, 327–355.

Griffin, R. W. & Moorhead, G. 2014. *Organizational Behavior: Managing People and Organizations*, 11th edition. Cengage Learning.

Habermas, J. 1985. *The Theory of Communicative Action, Volume 1: Reason and the Rationalization of Society*. Beacon Press.

Havelund, K. & Rosu, G. 2001. Monitoring programs using rewriting. In *Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE'01)*, 135–143.

Haynes, C., Miles, S. & Luck, M. 2014. Monitoring the impact of norms upon organisational performance: A simulation approach. In *Proceedings of the 9th International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN'13)*, *LNCS* 8386, 103–119. Springer.

Helbing, D. & Pournaras, E. 2015. Society: build digital democracy. *Nature* **527**, 33–34.
Hesselius, P., Johansson, P. & Vikström, J. 2008. *Monitoring and norms in sickness insurance: empirical evidence from a natural experiment*. Technical Report, 2008:8. Institute for Labour Market Policy Evaluation, Uppsala, Sweden.
Hoffman, E., McCabe, K. A. & Smith, V. L. 2007. Behavioral foundations of reciprocity: experimental economics and evolutionary psychology. *Economic Inquiry* **36**, 335–352.
Hogg, M. & Vaughan, G. 2014. *Social Psychology*, 7th edition. Pearson/Prentice Hall.
Hollander, C. D. & Wu, A. S. 2011. The current state of normative agent-based systems. *Journal of Artificial Societies and Social Simulation* **14**, 6.
Hübner, J. F., Sichman, J. S. & Boissier, O. 2002. MOISE+: towards a structural, functional, and deontic model for MAS organization. In *Proceedings of the 1st International Conference on Autonomous Agents & Multiagent Systems (AAMAS'02)*, 501–502. IFAAMAS.
Interis, M. 2011. On norms: a typology with discussion. *American Journal of Economics and Sociology* **70**, 424–438.
Janssen, M. A. & Ostrom, E. 2014. Vulnerability of social norms to incomplete information. In Xenitidou and Edmonds, (2014), 161–173.
Jennings, N. R. 1993. Commitments and conventions: the foundation of coordination in multi-agent systems. *Knowledge Engineering Review* **8**, 223–250.
Jiang, J., Dignum, V., Aldewereld, H., Dignum, F. & Tan, Y. 2013. Norm compliance checking. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, 1121–1122.
Kafal, Ö. & Torroni, P. 2012. Exception diagnosis in multiagent contract executions. *Annals of Mathematics and Artificial Intelligence* **64**, 73–107.
Kafal, Ö. & Torroni, P. 2018. Comodo: collaborative monitoring of commitment delegations. *Expert Systems with Applications* **105**, 144–158.
Kafal, Ö. & Yolum, P. 2016. PISAGOR: A proactive software agent for monitoring interactions. *Knowledge and Information Systems* **47**, 215–239.
Kalia, A. K., Nezhad, H. R. M., Bartolini, C. & Singh, M. P. 2013. Monitoring commitments in people-driven service engagements. In *Proceedings of the 10th IEEE International Conference on Services Computing (SCC'13)*, 160–167. IEEE.
Kameda, T., Takezawa, M. & Hastie, R. 2003. The logic of social sharing: An evolutionary game analysis of adaptive norm development. *Personal Social Psychology Review* **7**, 2–19.
Kelsen, H. 1991. *General Theory of Norms*. Clarendon Press.
Kemmerer, R. & Vigna, G. 2002. Intrusion detection: a brief history and overview. *IEEE Computer* **35**, 27–30.
Kirrane, S., Mileo, A. & Decker, S. 2017. Access control and the resource description framework: a survey. *Semantic Web* **8**, 311–352.
Kishon, A., Consel, C. & Hudak, P. 1991. Monitoring semantics: a formal framework for specifying, implementing and reasoning about execution monitors. In *Proceedings of the 4th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'91)*, 338–352. ACM.
Kowalski, R. A. & Sergot, M. J. 1985. A logic-based calculus of events. In *Foundations of Knowledge Base Management (Xania Workshop 1985)*, 23–55. Springer.
Lewis, D. 1969. *Convention: A Philosophical Study*. Harvard University Press.
Ligatti, J., Bauer, L. & Walker, D. 2005. Edit automata: enforcement mechanisms for run-time security policies. *International Journal of Information Security* **4**, 2–16.
Ligatti, J., Bauer, L. & Walker, D. 2009. Run-time enforcement of nonsafety policies. *ACM Transactions on Information Systems Security* **12**, 19:1–19:41.
Lorini, E. & Mühlenbernd, R. 2018. The long-term benefits of following fairness norms under dynamics of learning and evolution. *Fundamenta Informaticae* **158**, 121–148.
Luckham, D. 2002. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley.
Ly, L. T., Maggi, F. M., Montali, M., Rinderle-Ma, S. & van der Aalst, W. M. P. 2015. Compliance monitoring in business processes: functionalities, application, and tool-support. *Information Systems* **54**, 209–234.
Mahmoud, M. A., Ahmad, M. S., Yusoff, M. Z. M. & Mustapha, A. 2014. A review of norms and normative multiagent systems. *The Scientific World Journal* **2014**, 684587.
Mathis, R. L. & Jackson, J. H. 2010. *Human Resource Management*, 13th edition. Cengage Learning.
Modgil, S., Faci, N., Meneguzzi, F. R., Oren, N., Miles, S. & Luck, M. 2009. A framework for monitoring agent-based normative systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, 153–160. IFAAMAS.
Modgil, S., Oren, N., Faci, N., Meneguzzi, F., Miles, S. & Luck, M. 2015. Monitoring compliance with e-contracts and norms. *Artificial Intelligence and Law* **23**, 161–196.
Montali, M., Chesani, F., Mello, P. & Torroni, P. 2010. Monitoring time-aware social commitments with reactive event calculus. In *Proceedings of the 20th European Meeting on Cybernetics and Systems Research (EMCSR'2010)*, 447–452. Austrian Society for Cybernetic Studies.

Montali, M., Pesic, M., van der Aalst, W. M. P., Chesani, F., Mello, P. & Storari, S. 2010. Declarative specification and verification of service choreographiess. *ACM Transactions on the Web*, **4**, 3:–3:62.

Montali, M., Torroni, P., Chesani, F., Mello, P., Alberti, M. & Lamma, E. 2010. Abductive logic programming as an effective technology for the static verification of declarative business processes. *Fundamenta Informaticae* **102**, 325–361.

Morales, J., López-Sánchez, M., Rodrguez-Aguilar, J. A., Vasconcelos, W. W. & Wooldridge, M. 2015. Online automated synthesis of compact normative systems. *ACM Transactions on Autonomous and Adaptive Systems* **10**, 2:1–2:33.

Noriega, P., Chopra, A. K., Fornara, N., Cardoso, H. L. & Singh, M. P. 2013. Regulated MAS: Social Perspective. In (Andrighetto et al., 2013), 93–133.

Oracle, Inc. 2002. Triggers and stored procedures. In *Oracle Migration Workbench Reference Guide for Microsoft SQL Server and Sybase Adaptive Server Migrations Release* 9.2.0 for Microsoft Windows 98/2000 and Microsoft Windows NT, chapter 3. `https://docs.oracle.com/cd/B10501_01/win.920/a97248/ch3.htm`.

Ostrom, E. 1990. *Governing the Commons*. Cambridge University Press.

Ostrom, E. 2014. Collective action and the evolution of social norms. *Journal of Natural Resources Policy Research* **6**, 235–252.

Panagiotidi, S., Álvarez-Napagao, S. & Vázquez-Salceda, J. 2013. Towards the norm-aware agent: bridging the gap between deontic specifications and practical mechanisms for norm monitoring and norm-aware planning. In *Proceedings of the 9th International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN'13)*, *LNCS* 8386, 346–363. Springer.

Pesic, M. & van der Aalst, W. M. P. 2006. A declarative approach for flexible business processes management. In *Proceedings of the BPM'06 Workshops*, *LNCS* 4103, 169–180. Springer.

Pitt, J., Schaumeier, J. & Artikis, A. 2012. Axiomatization of socio-economic principles for self-organizing institutions: concepts, experiments and challenges. *ACM Transactions on Autononomous Adaptative Systems* **7**, 39:1–39:39.

Robertson, D. 2004. A lightweight coordination calculus for agent systems. In *Proceedings of the 2nd International Workshop on Declarative Agent Languages and Technologies (DALT'04)*, *LNCS* 3476, 183–197. Springer.

Savarimuthu, B. T. R. & Cranefield, S. 2011. Norm creation, spreading and emergence: a survey of simulation models of norms in multi-agent systems. *Multiagent and Grid Systems* **7**, 21–54.

Schneider, F. B. 2000. Enforceable security policies. *ACM Transactions on Information Systems Security* **3**, 30–50.

Schneider, F. B. 2018. Impediments with policy interventions to foster cybersecurity. *Communications of the ACM* **61**, 36–38.

Schnoebelen, P. 2002. The complexity of temporal logic model checking. In *Advances in Modal Logic*, Balbiani, P., Suzuki, N.-Y., Wolter, F. & Zakharyaschev, M. (eds). World Scientific, 1–44.

Seinen, I. & Schram, A. 2006. Social status and group norms: indirect reciprocity in a repeated helping experiment. *European Economic Review* **50**, 581–602.

Shoham, Y. & Tennenholtz, M. 1995. On social laws for artificial agent societies: off-line design. *Artifical Intelligence* **73**, 231–252.

Singh, M. P. 1998. Agent communication languages: rethinking the principles. *IEEE Computer* **31**, 40–47.

Singh, M. P. 1999. An ontology for commitments in multiagent systems. *Artificial Intelligence and Law* **7**, 97–113.

Singh, M. P., Arrott, M., Balke, T., Chopra, A. K., Christiaanse, R., Cranefield, S., Dignum, F., Eynard, D., Farcas, E., Fornara, N., Gandon, F., Governatori, G., Dam, H. K., Hulstijn, J., Krueger, I., Lam, H.-P., Meisinger, M., Noriega, P., Savarimuthu, B. T. R., Tadanki, K., Verhagen, H. & Villata, S. 2013. The Uses of Norms. In (Andrighetto et al., 2013), 191–229.

Srour, F. J. & Yorke-Smith, N. 2016. Assessing maritime customs process re-engineering using agent-based simulation. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'16)*, 786–795. IFAAMAS.

Stolz, V. & Bodden, E. 2006. Temporal assertions using AspectJ. *Electronic Notes in Theoretical Computer Science* 144, 109–124. Proceedings of the 5th International Workshop on Runtime Verification (RV'05). `https://www.sciencedirect.com/science/article/pii/S1571066106003069`.

Subrahmanian, V. S., Bonatti, P. A., Dix, J., Eiter, T. R., Kraus, S., Ozcan, F. & Ross, R. B. 2000. *Heterogeneous Agent Systems*. MIT Press.

Testerink, B., Bulling, N. & Dastani, M. 2015. Security and robustness issues in collaborative runtime verification. In *Proceedings of the 11th International Workshop on Coordination, Organizations, Institutions, and Normes in Agent Systems (COIN'15@IJCAI)*, 376–395. Springer.

Testerink, B., Dastani, M. & Meyer, J. C. 2014. Norm monitoring through observation sharing. In *Proceedings of the 2014 European Conference on Social Intelligence (ECSI'14)*, 1283, 291–304. CEUR Workshop Proceedings.

Thati, P. & Rosu, G. 2005. Monitoring algorithms for metric temporal logic specifications. *Electronic Notes on Theoretical Computer Science* **113,** 145–162.

Torroni, P., Chesani, F., Mello, P. & Montali, M. 2009. Social commitments in time: Satisfied or compensated. In *Proceedings of the 7th International Workshop on Declarative Agent Languages and Technologies (DALT'09)*, *LNCS* 5948, 228–243. Springer.

Torroni, P., Yolum, P., Singh, M. P., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E. & Mello, P. 2009. Modelling interactions via commitments and expectations. In *Handbook of Research on Multi-Agent Systems: Semantics and dynamics of organizational models*, Dignum, V. (ed), 263–284. Information Science Reference.

van der Aalst, W. M. P. & Pesic, M. 2006. DecSerFlow: towards a truly declarative service flow language. In *Proceedings of the 3rd International Workshop on Web Services and Formal Methods (WS-FM'06)*, *LNCS* **4184**, 1–23. Springer.

Vázquez-Salceda, J., Aldewereld, H. & Dignum, F. 2004. Implementing norms in multiagent systems. In *Proceedings of the 2nd German Conference on Multiagent System Technologies (MATES'04)*, *LNCS* 3187, 313–327. Springer.

Verhagen, H. 2001. Simulation of the learning of norms. *Social Science Computer Review* **19**(3), 296–306.

Villatoro, D., Sen, S. & Sabater-Mir, J. 2010. Of social norms and sanctioning: a game theoretical overview. *International Journal of Agent Technologies and Systems* **2**, 1–15.

von Wright, G. H. 1951. Deontic Logic. *Mind* **60**, 1–15.

Warnier, M., Dechesne, F. & Brazier, F. 2015. *Design for the Value of Privacy*, 431–445. Springer.

Westergaard, M. 2011. Better algorithms for analyzing and enacting declarative workflow languages using LTL. In *Proceedings of the 9th International Conference on Business Process Management (BPM'11)*, *LNCS* 6896, 83–98. Springer.

Widom, J. & Ceri, S. (eds) 1994. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann.

Xenitidou, M. & Edmonds, B. (eds) 2014. *The Complexity of Social Norms*. Springer.

Yolum, P. & Singh, M. P. 2002. Flexible protocol specification and execution: applying event calculus planning using commitments. In *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, 527–534.

Yolum, P. & Singh, M. P. 2004. Reasoning about commitments in the event calculus: an approach for specifying and executing protocols. *Annals of Mathematics and Artifical Intelligence* **42**, 227–253.

Young, H. P. 1993. The evolution of conventions. *Econometrica* **61**, 57–84.

Young, H. P. 2014. The evolution of social norms. *Economics Series Working Papers* 726. University of Oxford, Department of Economics.

Zuboff, S. 2015. Big other: Surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology* **30**, 75–89.