Noname manuscript No.
(will be inserted by the editor)

Monitoring Compliance with E-Contracts and Norms

Sanjay Modgil · Nir Oren · Noura Faci · Felipe Meneguzzi · Simon Miles · Michael Luck

the date of receipt and acceptance should be inserted later

Abstract The behaviour of autonomous agents may deviate from that deemed to be for the good of the societal systems of which they are a part. Norms have therefore been proposed as a means to regulate agent behaviours in open and dynamic systems, where these norms specify the obliged, permitted and prohibited behaviours of agents. Regulation can effectively be achieved through use of enforcement mechanisms that result in a net loss of utility for an agent in cases where the agent's behaviour fails to comply with the norms. Recognition of compliance is thus crucial for achieving regulation. In this paper, we propose a general framework for observation of agents' behaviour, and recognition of this behaviour as constituting, or *counting as*, compliance or violation. The framework deploys monitors that receive inputs from *trusted* observers, and processes these inputs together with transition network representations of individual norms. In this way, monitors determine the fulfillment or violation status of norms. The paper also describes a proof of concept implementation of the framework, and its deployment in electronic contracting environments.[1]

S.Modgil
King's College London E-mail: sanjay.modgil@kcl.ac.uk

Nir Oren
University of Aberdeen E-mail: n.oren@abdn.ac.uk

N.Faci
Claude Bernard University of Lyon 1 E-mail: noura.faci@liris.cnrs.fr

Felipe Meneguzzi
Pontifical Catholic University of Rio Grande do Sul E-mail: felipe.meneguzzi@pucrs.br

Simon Miles
King's College London E-mail: simon.miles@kcl.ac.uk

Michael Luck
King's College London E-mail: michael.luck@kcl.ac.uk

[1] Michael Luck gave a keynote talk at the Fifteenth International Conference on AI and Law, part of which was based on the work reported in this paper.

1 Introduction

Business interactions are typically mediated through the use of contracts, by which parties agree to provide goods or services to one another in support of overall business objectives. Here, contracts offer the guarantees that are needed to provide a degree of assurance to the contract parties, so that transactions can take place in a secure and committed context. In seeking to automate this by means of electronic business systems, one therefore requires some analogous focus on providing guarantees for service delivery. While there has been some previous work on such contract-based systems, in particular driven by work on norms and normative reasoning, this paper is primarily concerned with the development and deployment of practical systems for business scenarios. In particular, this paper addresses the issues that arise when seeking to provide assurance over the actions of others. This is achieved through the use of monitoring techniques that determine when a business agreement has been violated so that remedial action may be taken, and when it has been fulfilled so that the agreement concludes successfully. In doing so, the paper adopts a normative stance, seeing agreements or contracts as specified by norms that regulate system behaviour.

Against this background, multi-agent systems provide an ideal context in which to consider the problems raised by monitoring electronic business contracts, since they reflect the nature of self-interested, autonomous, problem-solving entities working together to achieve some overarching objective, while satisfying their own individual needs. Indeed, recent years have witnessed a growing interest in the use of *norms* to regulate and coordinate agent behaviours, and so achieve the overall objectives of multi-agent systems. Such norms specify the actions that an agent may, should, or should not undertake, and states of affairs within the environment that an agent may, should, or should not, allow to occur.

For example, consider the aerospace industry in which the behaviours of airline operators, engine manufacturers, and service sites are required to comply with (amongst others) norms governing the repair of engines and sourcing of parts for these repairs. Typically, engine manufacturers are under *obligation* to have operational engines available for the planes of a client airline operator. In order to meet such obligations, service sites (located at airports) are obliged to repair engines for engine manufacturers within a given time period. Other types of normative prescription include *permissions* and *prohibitions*. For example, a service site may either be permitted to, or prohibited from, sourcing parts for engine repair from certain part manufacturers (where these provenance restrictions can be inherited from the requirements of the client airline operator).

Two approaches have been taken in considering the use of norms in agent systems. In the *regimentation* approach [18], adopted for example by electronic institutions [9], agent behaviour is constrained to that specified by norms. Here, agent autonomy is drastically curtailed, and such regimented systems are less flexible in that only appropriately specified agents can join. In contrast, the *enforcement* approach [5, 7, 16, 21, 30] accommodates agents that preserve a degree of autonomy in order that they may behave in a more flexible, responsive, and ultimately more intelligent manner. Such autonomy implies that agents can violate norms if it is in their interest to do so, and therefore enforcement mechanisms are required to motivate agent compliance by threatening some loss of utility for agents in case of violation. The enforcement approach therefore requires that agent actions are

monitored; that is, they must be observable and recognised as complying with or violating norms, in order that the enforcement mechanisms may be appropriately applied.

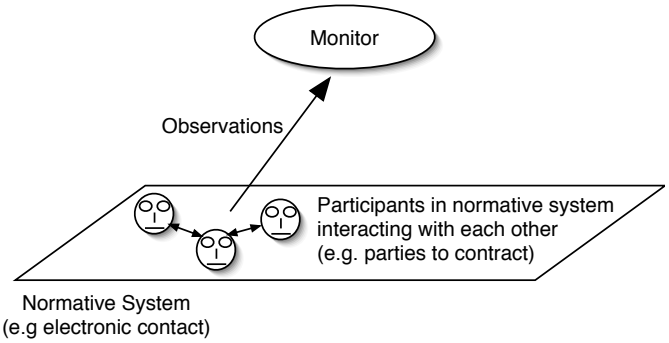

Fig. 1 Monitor processes transition network representations of norms in underlying normative system, together with observations of agents participating in normative system, in order to determine the status of norms.

In this paper, we adopt the enforcement approach, since it reflects the kind of situation we expect in business scenarios in which the participating entities are completely autonomous and can choose to violate norms. We enumerate a set of requirements that we argue should be met by any general and reusable framework for monitoring of normative multi-agent systems, and describe a framework for monitoring that satisfies these requirements, outlining a proof of concept implementation of the framework and its use in monitoring norms encoded as clauses in electronic contracts. Our framework describes the use of monitor agents for deployment in a range of normative systems (see Figure 1), and assumes a model [30] that abstracts from the specific representational formalism for encoding norms, and is thus to some extent normative system neutral. We describe how a normative system's individual norms can be represented as *transition networks* that may be used to match against observations to determine the current status of the norms, and to facilitate further action in case of violation. In particular, we introduce the notion of a *monitor* that can report on whether a norm has been fulfilled or violated, so that sanctions can be applied as and when appropriate. The transition networks used also provide for rudimentary explanations of normative violations. Two key features of the framework are that:

1. there is a requirement for explicit agreement between the normative system's participating agents as to what world features constitute violation or fulfilment of a norm, where these constitutive features can be directly mapped to transition network arc labels; and
2. the system's participating agents explicitly entrust *observers* to accurately observe and report the observed world features to *monitor* agents, and the participating agents entrust monitors to report accurately on any violations that occur.

This paper, which is a revised and extended version of [26], therefore makes the following two distinct research contributions. First, it enumerates requirements

for a general monitoring framework for normative multi-agent systems. Second, it formalises such a general monitoring framework meeting the above requirements.

The paper is organised as follows. Section 2 describes example monitoring scenarios that are referred to throughout the paper, and enumerates a set of requirements for monitoring in normative systems. Section 3 then describes some general normative concepts that our approach to monitoring makes use of; in particular the model of norms described in [30]. Section 4 then motivates and describes an architectural overview of our approach, with particular emphasis on the relationship between our normative framework and the underlying normative system, and how this relationship is partly established by agreements between the agents participating in the normative system. The remainder of Section 4 describes how individual norms are represented as transition networks, and processed by monitor agents, and how the status of a norm is evaluated and reported together with some limited explanation. Section 5 describes validation of our approach. We report on a proof of concept implementation of a monitoring agent, and its processing of transition network representations of norms encoded in an electronic contract [23, 24] specified by the CONTRACT project[2]. The implementation builds on work on electronic representations and software tools for contracts [30] and their use in a number of case studies [17]. The implementation demonstrates monitoring of *AgentSpeak(L)* agents [31] whose interactions are governed by normative clauses specified in an aerospace contract. In Section 6 we describe future work; in particular, how our approach can be extended to provide more comprehensive explanations, and to implement *predictive* monitoring (whereby a state can be recognised as one in which a norm is *in danger* of being violated). Section 7 discusses related work, and we conclude in Section 8.

2 Requirements for Monitoring

In this section, we enumerate those requirements that should be met by any general and reusable framework for monitoring of normative multi-agent systems. Broadly speaking, three distinct categories of requirements can be distinguished:

- The first category relates to requirements on participating agents to agree explicitly as to what world features constitute (*count as*) [32] fulfilment or violation of norms, and what entities can be trusted to accurately observe and report such features, and appropriately apply enforcement mechanisms.
- The second category describes requirements on monitors: to detect the *status* of any given norm (i.e., whether the norm applies to some agents at any given time so that it is in force), or whether it has been violated, or fulfilled or is no longer in force (expired); to inform participating agents of the norms that apply to them and when they have violated or fulfilled norms; and to provide proper explanations of violations of norms so that responsibility can be appropriately assigned.

  As discussed later, fulfilment of the above two categories of requirements ensures that a more general requirement is met: to motivate agent participation in normative systems requires some assurance that enforcement mechanisms, such as punishments or sanctions, are employed only as and when appropriate.

[2] `ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/enet/090219-contract_en.pdf`

– The third category relates to the adequacy of the monitoring framework's representational model of norms insofar as such a model should account for different types of norms of varying degrees of complexity. At the same time, it should also limit commitment to the specific representation of norms in the system being monitored so as to ensure (to the extent that it is possible) that the framework can be applied to a range of underlying normative systems so that it is *normative system neutral*.

We discuss each category of requirements in the subsections below.

2.1 Requirements on Agents Participating in Normative Systems

To motivate agent participation in normative systems requires that the participating agents are given some degree of assurance that enforcement mechanisms, such as punishments or sanctions, are employed only as and when appropriate. To illustrate, consider the example in Table 1, which describes a purchasing scenario involving a purchaser, P, who is buying goods, G, from a supplier, S. Here, the purchaser P can be broken down into two parties, the actual buyer, B, and the financial department, F, that is responsible for payment.

Clearly, any possibility that an agent (for example, representing the financial department, F) may be sanctioned for not complying with a norm when in actuality the agent has complied, will discourage such an agent from participating in the normative system. Conversely, any possibility that an agent (e.g., F) may *not* be sanctioned for violating a norm when in actuality the agent has violated it, will discourage participation of an agent (for example, the supplier of goods, S) who is disadvantaged as a result of the violation. The likelihood of both these types of scenario occurring to some extent depends on how violations of norms are recognised. In the scenario of Table 1, these amount to the following specific situations.

1. Suppose S's observation that monies have been deposited in S's bank account constitutes fulfilment of the obligation on F. This is open to abuse in that S may not inform a monitor that the monies have been deposited, resulting in some inappropriate sanction on F (that in turn benefits S).
2. Conversely, suppose that the obligation on F is deemed to be satisfied by a monitor agent if a message is observed as having been sent from F to S, informing the latter that the monies have been paid. This is clearly open to abuse, in that if F does not actually pay, F can still send the message (where observation of this message will indicate fulfilment) and thus avoid sanction.

Hence, to motivate participation of agents in normative systems requires that the agents explicitly agree as to what constitutes violation and fulfilment of a norm. We can understand this requirement in terms of Searle's work on constitutive rules and socially constructed (institutional) facts [32]; collective agreement as to the constitutive X *counts as* Y is needed, where the X term is the brute fact (observation) that counts as the institutional fact that is the Y term (the normative violation or fulfilment). Agents can thus agree as to what constitutes violation or fulfilment of a norm, *as well as* when a norm is active (in force) and when it has expired (no longer in force). They can thus limit opportunities for

Consider an example norm, which we label *NormGoods*, and which describes an obligation on the purchaser P of goods G from a supplier S, where the purchaser P is an organisational entity consisting of two contractual parties: the buyer B and the financial department F.

If buyer B is notified by S that goods G are in stock then, unless S is declared bankrupt, **either**
- B must cancel the order within 7 days of receipt of notification, **or**;
- B must accept the order **and** F must pay S for goods G within 7 days of receipt of notification.

Table 1 Example: **Goods Obligation Example**

abuse by participating agents. In particular, it is the *recorded* existence of such explicit agreements that help to forestall opportunities for contesting application of sanctions (for example, through litigation).

Suppose now that S and F agree that the presence of monies deposited in S's bank constitutes fulfilment of the obligation. This precludes the type of abuse in the second situation, which we refer to as *sanction avoidance*. However, it does not preclude abuse of the type described in the first situation, which we refer to as *sanction imposition*. To preclude the latter additionally requires that a *trusted* observer is responsible for both observing the presence of the monies in the bank account, and relaying this observation to a monitoring agent. Furthermore, participating agents also require assurances that observations are appropriately processed and that the status of a given norm is appropriately reported by monitor agents. In summary, the following two requirements on agents in normative systems are important in order to motivate agent participation in the normative system being monitored.

R1 *A monitoring framework applied to a normative system, $\mathcal{NS}$, requires agreement among agents participating in $\mathcal{NS}$ as to* what *features of the world constitute fulfilment or violation of norms.*

R2 *A monitoring framework applied to a normative system, $\mathcal{NS}$, requires agreement among agents participating in $\mathcal{NS}$ as to* who *is trusted to accurately observe and report the above features.*

2.2 Requirements on the Monitoring Framework

So far, we have elaborated requirements that a monitoring framework should impose on the normative system that it monitors. However, additional requirements on the monitoring framework itself are also related to assurances that enforcement mechanisms are employed only as and when appropriate. In particular, there should be detection and reporting of a given norm violation, and proper analysis of normative violations so as to ensure that responsibility for violation is properly assigned, and that mitigating circumstances are recognised. This means that reporting violation of a norm must be accompanied by explanations that permit *diagnosis*. Such diagnostic explanations may also help ensure that (remedial) changes to normative specifications can be appropriately made, so as to ensure that the exceptional circumstances are accounted for, and so violations are less frequent.

Consider the example norm *DriveLeft*, that describes an obligation on a given agent to drive on the left.
If the agent is driving, then it must drive on the left.

Table 2 Example: **Driving Obligation Example**

R3 *A monitoring framework should detect and report on when a norm is violated, and provide mechanisms for explanation of norm violations.*

In order to maximise chances of compliance, agents must be made aware of when norms apply to them, and the sanctions that will be imposed in case of violation (so that the threat of sanction can have the required motivational force). Ideally, agents should also be informed of when they are in danger of violation, so that they may take appropriate measures. Any approach to monitoring should therefore recognise and report not only on when norms are violated (or fulfilled), but also on the *states* in which norms come into force (and possibly other states in which norms are in danger of being violated) and when norms are no longer in force. The results of such monitoring can then be fed back to the agents.

R4 *A monitoring framework should detect and report on when a norm comes into force, when a norm is not in force, and inform agents of possible sanctions in case of violation.*

2.3 Requirements for Representation of Norms

Norms may be represented and implemented in many different ways, and in different contexts. For example, norms may be represented as logical formulae in deontic logic contexts (e.g., [34]), as clauses in electronic contracts (e.g., [30]), or as programming constructs in programming environments for normative multi-agent systems (e.g., [7]). Thus, for a monitoring framework to be applicable to a wide variety of normative systems requires that the representation of norms assumed by the framework is (to the extent that it is possible) *normative system neutral*. That is, the representation of norms for processing by monitors should not commit to specific representational and implementation features, nor to *dependencies* between norms (since one would not want to assume any workflow commitments encoded by these dependencies in the underlying normative system). We thus specify the following requirement.

R5 *Modular representations of norms should be available for monitoring, where these representations do not commit to specific representation of, or inter-dependencies between, norms in the normative system being monitored.*

Norms specify behaviours and world states that are obliged, permitted and prohibited, and that apply to agents acting jointly. These agent behaviours and world states may require complex representations, rather than simple atomic logical representation. For example, consider the *Goods Obligation* in Table 1 that applies to both the buyer B and the financial department F, where what is obliged is specified as a disjunction, where the second disjunct is itself a conjunction.

Furthermore, norms not only identify states that must be realised (achieved) at a given moment in time, as in Table 1, but also states that must (or may or

must not) be *maintained* over a given time period. These are respectively referred to as *achievement* norms and *maintenance* norms. For example, Table 2 describes a scenario with a maintenance obligation, in which an agent must always drive on the left[3]. Here, the status of the obligation can toggle between violated (whenever the agent is driving on the right) and fulfilled (whenever the agent is driving on the left) during the period for which the norm is in force (that is, while the agent is driving). We thus identify the following requirements.

R6 *Any general and widely applicable approach to monitoring must account for representation of complex behaviours and states of interest, enacted and brought about jointly by groups of agents.*

R7 *Any general and widely applicable approach to monitoring must account for both achievement* and *maintenance obligations.*

In the subsequent sections we describe a general framework for monitoring and discuss how the framework satisfies thes above listed requirements. We begin by reviewing previous work [30] on a general model of norms that is sufficiently abstract as to be applicable in a variety of normative contexts. The model makes some conceptual distinctions that are of particular relevance from a monitoring perspective, and that we thus adopt for this paper (although the framework can easily assume other models).

3 A General Model of Norms

In the model described in [30], some general normative concepts shared by existing work on norms and normative systems (such as [11,19]) are identified. This model distinguishes between different types of norms: obligations, prohibitions and permissions. In this paper, our primary focus is on obligations, given that our main interest is in monitoring obligations and prohibitions, and that [30] models prohibitions to do X (or bring about X) as obligations not to do X (not to bring about X). However, we also consider *permissions* and will illustrate monitoring of obligations *and* permissions in Section 5's use case.

More specifically, the model identifies whether the norm is an obligation or permission (the $NormType$). It also distinguishes under which conditions the norm comes into force ($NormActivation$), the state of interest ($NormCondition$) obliged or permitted to be brought about by the agents to which the norm is addressed ($NormTarget$), and the conditions under which the norm is no longer in force ($NormExpiration$). (We refer to $NormActivation$, $NormCondition$ and $NormExpiration$, collectively, as a norm's *components*.) Thus, a norm $\mathcal{N}$ is modelled as a tuple:

$\langle NormType$
$NormActivation,$
$NormCondition,$
$NormExpiration,$
$NormTarget\rangle$

[3] One can conceive of this obligation as applying to human and automated agents (robot vehicles).

An instance of $\mathcal{N}$ is said to come into force, or is *activated*, if the conditions, or *state of interest*, described by $NormActivation$ hold. When $\mathcal{N}$ is activated, then $\mathcal{N}$ is *not violated* if the state of interest described by $NormCondition$ is brought about by $\mathcal{N}$'s $NormTarget$ in the case that $\mathcal{N}$'s $NormType$ is *obligation*; similarly, when $\mathcal{N}$ is activated, then $\mathcal{N}$ is *executed* if the state of interest described by $NormCondition$ is brought about by $\mathcal{N}$'s $NormTarget$ in the case that $\mathcal{N}$'s $NormType$ is *permission*. The norm remains in force until such a time as the state described by $NormExpiration$ holds.

The states of interest referred to above describe states of the world in which actions have been performed (for example, messages sent) or certain properties hold (for example, *the temperature is maintained above 23 degrees for at least 90% of the time*). [30] build on this model to develop an operational semantics for normative systems, whereby one can reason about norms and their changing status over time.

To illustrate this general model, the example norms of Table 1 and 2 are represented in the appropriate structure in Table 3 and Table 4. For the Goods Obligation of Table 3, the final clause of $NormCondition$ indicates that the obligation is not violated as long as the current time is within seven days of receipt of notification that the goods are in stock. If the seven day period elapses, and it does not hold that B has cancelled the order, or B has accepted the order and F has paid S for goods G, then the obligation is said to be violated.

Notice that if S is bankrupt, as indicated in the final clause of $NormExpiration$, then the norm no longer applies. While such an exception might be expected to have been encoded in the activation condition, if so encoded, it may be that S is declared bankrupt *after* the norm has been activated, and so the norm would inappropriately remain in force. Encoding this exception in the expiration condition ensures that the norm ceases to be in force in such circumstances.

Table 3 Goods Obligation

$NormType$	obligation
$NormActivation$	buyer B is notified by S that goods G are in stock at time T
$NormCondition$	B cancels the order, or B accepts the order and F pays S for goods G, or it is less than 7 days after T
$NormExpiration$	B cancels the order, or B accepts the order and F pays S for goods G, or it is greater than 7 days after T, or S is bankrupt
$NormTarget$	B, F

Table 4 Driving Obligation

$NormType$	obligation
$NormActivation$	an agent X begins driving
$NormCondition$	agent X is driving on the left
$NormExpiration$	agent X stops driving
$NormTarget$	X

4 A Framework for Monitoring

Given the model of norms just introduced, we can now proceed to describe a framework for monitoring the behaviour of agents deployed in normative systems, highlighting how the framework addresses the requirements enumerated in Section 2. Recall that we aim at an approach that is generic and applicable to a range of dynamic open normative systems, including normative *organisations* developed by the kinds of dedicated languages described in [7], as well as electronic contracting frameworks [30] in which contract clauses specify norms that the contract parties must comply with.

4.1 Monitoring Framework Architecture

We begin by describing the monitoring framework architecture: the relationships and information flows between a normative system's constituent agents, the various entities responsible for observing, monitoring and managing norms, and the environment. The architecture is so specified as to provide for satisfaction of requirements R1 and R2 from Section 2.1.

The architecture (shown in Figure 2) includes trusted observers that report to monitors on whether states of interest referenced by norm components do or do not hold. A monitor (which is itself an agent, and represented by the large oval in the centre of the figure) processes these observations together with transition network representations of norms (described in detail in Section 4.3) to determine whether a violation (for example) has occurred. Agents are treated as black boxes so that their internal state transitions are invisible to the monitors; the only assumptions we make about the normative system, $\mathcal{NS}$, being monitored, and the agents deployed in $\mathcal{NS}$ are as follows.

1. The norms in $\mathcal{NS}$ conform to the general model of norms in Section 3, in the sense that it is possible to identify a norm's type, target, and components ($NormActivation$, $NormCondition$ and $NormExpiration$).
2. The agents in $\mathcal{NS}$ are capable of making agreements as to what features of the world *count as* a given norm's components.
3. The agents in $\mathcal{NS}$ are capable of making agreements as to which entities are responsible for observing and reporting the features in (2), and monitoring the norms.

A *mapper* maps norms contained in $\mathcal{NS}$ to their network representations. These mappings also take as input the contents of the agreements described in point (2) above, where these contents are required for annotation of the network representations. The network representations are subsequently provided as off-line input to the monitors (identified by the agents in $\mathcal{NS}$ or other parties).

At run time, *monitors* subscribe to all observers entrusted with reporting on the states of interest identified by a norm's components. These monitors can identify which observers to subscribe to, based on the network representations. Notice that there is nothing in the specification of the monitor that ties it to a particular normative system.

Observers notify monitors as to whether states of interest hold, by notifying monitors of predicates describing properties of the world. These properties may

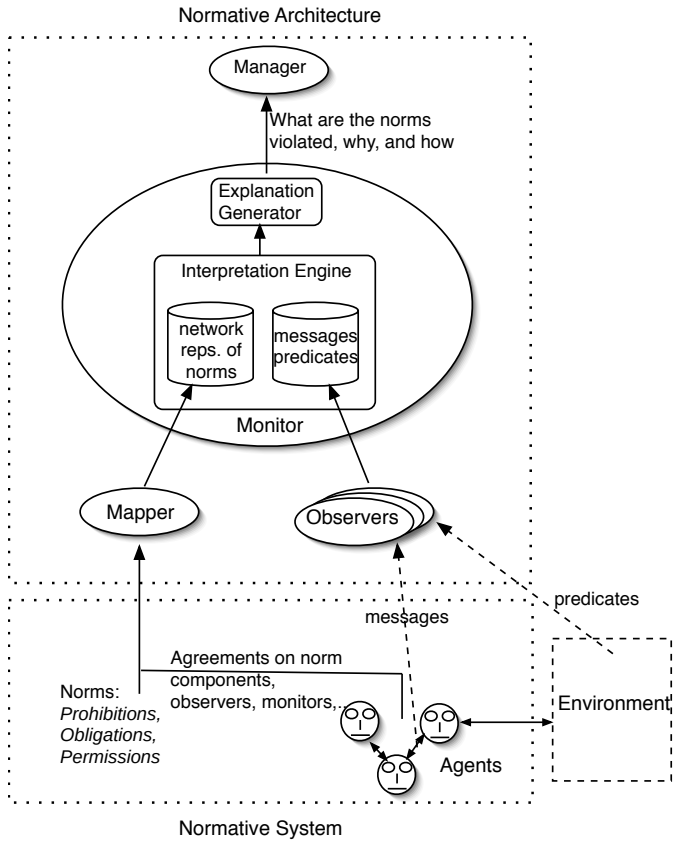

Fig. 2 Monitoring architecture and its relationship to a normative system.

refer to actions having been performed, where such actions include messages exchanged among agents and messages exchanged between agents and the environment. The observers are external to the normative system itself; their role is only to report on whether predicates hold, and they are not responsible for any kind of processing of this information. Thus, any environmental artefact can be assigned trusted observer status, including internet sites, human agents, banks, description logic reasoners, etc.

Monitors process observations together with the network representations of the norms, to determine when a norm is activated, fulfilled or violated, or has expired. Finally, the monitor informs manager agents of norms that have been violated, and of the agents responsible for violation. Manager agents then, in turn, impose sanctions on the relevant agents.

To reiterate, the choice of observers (and monitors) is application specific, and agreed to by the agents whose behaviours are being observed, where such agreements constitute declarations of trust. However, the behaviours of observers (and monitors) can themselves be governed by normative clauses, and thus observed and monitored for deviation from their expected behaviour. This would reduce the potential for collusion (for example, an agent dealing with eBay is more likely

to trust a PayPal observer, even though PayPal is owned by eBay, if the behaviour of PayPal is itself normatively prescribed and sanctioned in case of violation).

4.2 An Overview of Norm Representation and Processing

In this section we provide an overview of how individual norms are represented and processed by monitors. We summarise how our approach satisfies the requirements enumerated in Sections 2.2 and 2.3. In Section 4.3 we then more formally describe how norms conforming to Section 3's semantic model are mapped to *transition networks*. Section 4.4 presents a monitoring algorithm for processing such *transition networks*.

4.2.1 Transition network representations of norms

Individual norms — *obligations* and *permissions*[4] — can be represented as transition networks[5] that conform to the semantic model reviewed in Section 3. According to the model, an obligation norm is *abstract* until it is instantiated, and it is *activated* if the condition specified by $NormActivation$ holds, at which point it may be *violated* or *not violated* depending on whether the $NormCondition$ holds. The norm remains in one of these latter two states, potentially switching between them, until it *expires* when $NormExpiration$ holds. Given this model, and these states, appropriate transition networks are thus labelled directed graphs of the form:

$$(\{S1, S2, S3, S4, S5\}, \mathcal{A}\overrightarrow{12}, \mathcal{A}\overrightarrow{23}, \mathcal{A}\overrightarrow{24}, \mathcal{A}\overrightarrow{34}, \mathcal{A}\overrightarrow{43}, \mathcal{A}\overrightarrow{35}, \mathcal{A}\overrightarrow{45})$$

where, for $i, j = 1 \ldots 5$, Si is a node, and $\mathcal{A}\overrightarrow{ij}$ denotes a set of labelled arcs connecting Si to Sj. Figure 3 depicts a generic transition network representation of a norm where, intuitively, $S1$ denotes that the norm is abstract, $S2$ denotes that the norm is instantiated (i.e., activated or in force), $S3$ denotes that the norm is violated, $S4$ denotes that the norm is not violated, and $S5$ denotes that the norm has expired (is no longer in force).

A monitor's processing of such a transition network involves matching observations relayed to the monitors by trusted observers. These observations describe the states of interest specified by the norm's components, $NormActivation$, $NormCondition$ and $Norm\ Expiration$. The monitor matches these observations against the labels of the transition network's arcs indicating the corresponding states of interest, so as to transition the network from one node to the next. In this way, a monitor can determine when a norm becomes activated, expires, and when it fulfilled or violated.

[4] Recall that we assume *prohibitions* to do X (or bring about X) are modelled as obligations not to do X (bring about X).

[5] Note that these network representations share some features in common with Augmented Transition Networks (ATNs) [35], where the latter provide for recursive labelling of arcs by ATNs themselves (reflecting their initial development for natural language processing).

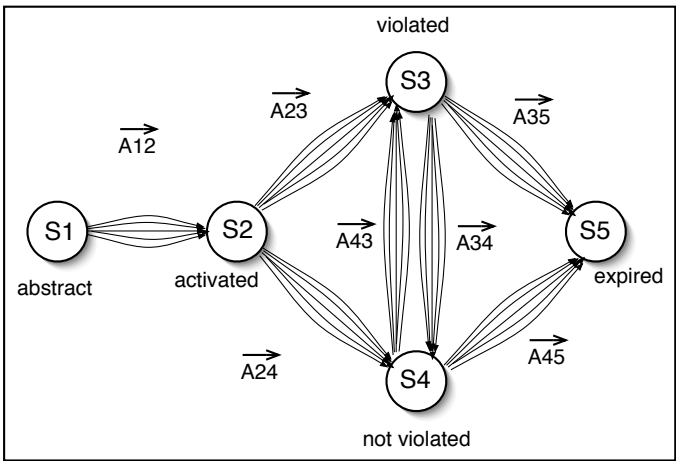

Fig. 3 Generic *transition network* representation of a norm.

4.2.2 Processing transition networks

More specifically, each transition indicated above corresponds to a particular meaningful transition in the status of a norm. In this subsection, we elaborate these specific transitions in more detail, for each norm component.

In what follows we assume that each norm component is represented in disjunctive normal form; that is, each norm component is of the form, $\alpha_1 \vee \alpha_2 \vee \ldots$, where each α_i is a conjunction, $\beta_1 \wedge \beta_2 \wedge \ldots$, and each β_j is a possibly negated atomic predicate formula, or complex temporal expression.

Consider an abstract norm $\mathcal{N}$. When instantiated as $\mathcal{NI}$, and so activated due to $\mathcal{N}$'s activation condition holding, a copy of the transition network representing $\mathcal{N}$ is made, where the copy has transitioned across one of the arcs in $\mathcal{A}\overrightarrow{12}$ so that the transition network for $\mathcal{NI}$ is in the activation state $S2$. When the transition network $TN_{\mathcal{N}}$ representing a norm $\mathcal{N}$ is copied to obtain the transition network $TN_{\mathcal{NI}}$ representing $\mathcal{NI}$, we say that $TN_{\mathcal{NI}}$ is an *instantiation* of $TN_{\mathcal{N}}$.

The arcs in $\mathcal{A}\overrightarrow{12}$ are labelled by the state of interest identified by $NormActivation$ where, as indicated above, $NormActivation$ is of the form $\alpha_1 \vee \alpha_2 \vee \ldots$, and each arc is labelled by one of these disjuncts (where each disjunct may itself be a conjunction). Hence, if observers send messages to a monitor indicating that at least one α_i holds, then the monitor transitions the corresponding arc of $TN_{\mathcal{N}}$, resulting in the instantiated $TN_{\mathcal{NI}}$ being in $S2$. Thus not only can the monitor report that the norm $\mathcal{N}$ is activated, but it can also report the reasons for the activation (that α_i holds).

Now, the instantiated norm $\mathcal{NI}$ must either be violated or not violated. In the latter case, we use $\mathcal{A}\overrightarrow{24}$ for the relevant transitions. The arcs in $\mathcal{A}\overrightarrow{24}$ are labelled by the state of interest identified by $NormCondition$, where $NormCondition = \gamma_1 \vee \gamma_2 \vee \ldots$, and each arc is labelled by one of these disjuncts. Hence, if upon activation, observers send messages to the monitor indicating that at least one γ_i holds, then the monitor transitions the corresponding arc in $\mathcal{A}\overrightarrow{24}$, so that $TN_{\mathcal{NI}}$ is now in the state $S4$ in which:

– if $NormType$ = permission then $\mathcal{NI}$ is said to have been made use of (executed); and
– if $NormType$ = obligation then $\mathcal{NI}$ is said to be not violated.

As indicated above, not only can the monitor report the status of $\mathcal{NI}$, but it can also provide the *reasons* as given by the label of the arc transitioned.

Conversely, we can consider the case in which a norm is violated. Suppose $NormCondition$ is of the form $(\beta_1 \wedge \beta_2) \vee (\beta_3)$. Then, by De Morgan's laws $NormCondition$ does not hold if neither β_1 or β_3 hold (despite the fact that β_2 may hold), or neither β_2 or β_3 hold (despite the fact that β_1 may hold). Thus, (as is made more precise in Section 4.3) $NormCondition$ defines the labels of arcs in $\mathcal{A}\overrightarrow{23}$, such that if immediately upon activation, the state of interest identified by at least one arc in $\mathcal{A}\overrightarrow{23}$ does not hold (in which case we say that *$NormCondition$ does not hold*), then $TN_{\mathcal{NI}}$ transitions across this arc to $S3$, where:

– if $NormType$ = permission then $\mathcal{NI}$ is said not to have been made use of (not executed); and
– if $NormType$ = obligation then $\mathcal{NI}$ is said to be violated.

Notice that a permission may toggle between not executed and executed, and an obligation may toggle between violated and not violated. The latter may occur when we are dealing with a maintenance obligation such as *always drive on the left*; the obligation may toggle between violated and not violated depending on whether the driver is driving on the right or the left at any given time point. Similarly a permission to drive on the left may or may not be executed at any given time point. In general then, if $TN_{\mathcal{NI}}$ is in $S3$, and $NormCondition$ holds, then the norm transitions from $S3$ to $S4$. If $TN_{\mathcal{NI}}$ is in $S4$, and $NormCondition$ does not hold, then the network transitions to $S3$.

Finally, we need to consider the case of norm expiration. If some disjunct in $NormExpiration = \epsilon_1 \vee \epsilon_2 \vee \ldots$ holds at the time of activation, then the transition network for the abstract norm is not instantiated. If, on the other hand, we already have the instantiated transition network, ($TN_{\mathcal{NI}}$), and observers send messages to the monitor indicating that at least one ϵ_i holds then, if $TN_{\mathcal{NI}}$ is in $S3$ or $S4$, the monitor transitions the corresponding arc, so that $TN_{\mathcal{NI}}$ is now in the expired state $S5$, and the monitor can report: first that the norm has expired, providing the reasons as given by the label of the arc transitioned; and second whether the norm expired having been fulfilled (executed) if the transition to $S5$ was from $S4$, or violated (not executed) if the transition to $S5$ was from $S3$.

4.2.3 Example transition networks

We illustrate these transitions with some example transition network representations of norms and their processing. Consider the maintenance driving obligation from Section 3. Informally, this norm can be represented as the transition network in Figure 4. Now, if an observer informs the monitor that *driver X* has set out on a car journey at time $T1$, then the instantiated network is transitioned across to $S2$. Now suppose that an observer informs the monitor that *driver X* has set out *not* driving on the left (and so driving on the right) at time $T1$. Hence, on the same time tick the network transitions to the *violated* state $S3$. If, at the next time tick $(T1 + 1)$, *driver X* is observed as driving on the left, then the network transitions

to the *non violated* state $S4$. It can be seen that the *transition network* can toggle between $S3$ and $S4$ depending on whether driver X is observed as driving on the right or left, until such a time as the expiration condition holds (and *driver X* is observed as having ended his journey).

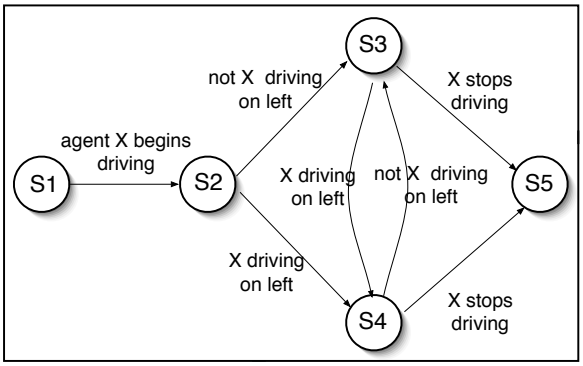

Fig. 4 Informal illustration of a maintenance obligation represented as a *transition network*

Table 5 Traffic Warden Obligation

$NormType$	obligation
$NormActivation$	car is parked on a yellow line between 2pm and 3pm
$NormCondition$	either post a penalty notice before leaving the scene, or call for a tow truck before leaving the scene
$NormExpiration$	a penalty has been posted, or a tow truck has been called, or the warden has left the scene
$NormTarget$	traffic_warden

As a second example, consider the achievement obligation in Table 5, in which the norm specifies what a traffic warden is obliged to do when a car parks on a yellow line between 2pm and 3pm.[6] Informally, this norm can be represented as the transition network in Figure 5. If an observer informs the monitor that $NormActivation$ holds true, then the instantiated network is created, and transitioned across to $S2$. Now, if an observer informs the monitor that the traffic warden does not immediately leave the scene, then the network transitions across the arc labelled *not left scene* to the node $S4$ denoting the *not violated* state of the norm. At this point, either one of the following situations arises.

– The warden posts a penalty notice. Here, $NormExpiration$ holds and the obligation has been fulfilled since it is in $S4$ prior to transitioning across the corresponding arc to the expiration state $S5$.

[6] Note that the fact that the norm does not expire after 3pm means that the traffic warden is still obliged to penalise as long it is the case that the car was observed on a yellow line in the 2-3pm time period.

– The warden does not post a penalty notice and does not call a tow truck, and leaves the scene. The conjuncts on the arc from $S4$ to $S3$ hold, and the arc is preferentially transitioned to $S3$ *and then to the expiration state* $S5$ along the arc labelled *left scene*. Notice the importance of the procedural requirement that the labels of arcs between $S3$ and $S4$ are checked and transitioned prior to the arcs leading from $S3$ and $S4$ to $S5$. In this example, it would be inappropriate to transition immediately from $S4$ to $S5$ along the arc labelled *left scene*, since the warden would then incorrectly be deemed to have fulfilled his obligation. Furthermore, notice that achievement obligations do not toggle from violated to not violated, so that the arcs in $\mathcal{A}\overrightarrow{34}$ are not (and logically cannot be) transitioned. However, for simplicity we assume a uniform transition network representation of maintenance and achievement obligations and permissions.

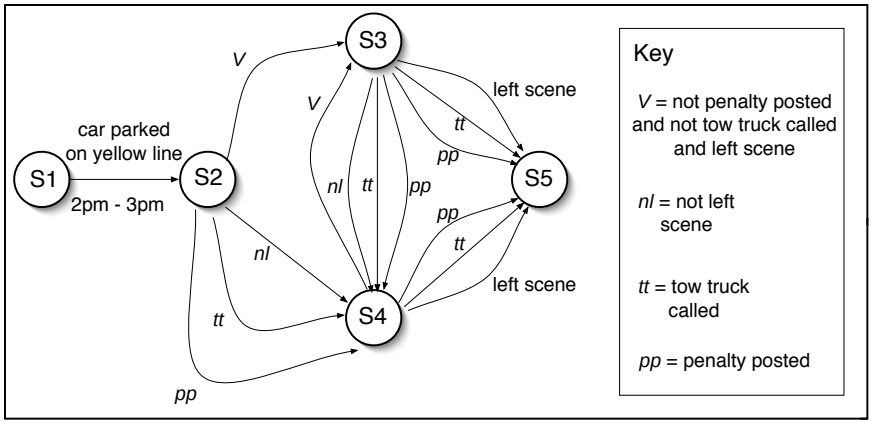

Fig. 5 Informal illustration of an achievement obligation represented as a *transition network*

4.2.4 Requirements on monitoring and norm representation

Given this model, it can be seen that the transition network representation of norms for monitoring satisfies the requirements R5, R6 and R7 enumerated in Section 2.3 given that:

1. it assumes an abstract general model of norms;
2. it provides for representation of complex behaviours and states of interest enacted and brought about jointly by groups of agents;
3. it models achievement *and* maintenance norms;
4. only behaviours specified by the norms are represented, so that a given transition network can represent the same norm specified in any one of a number of normative systems; and
5. transition network representations of norms are independent of each other, allowing run time addition and removal of norms.

Furthermore, the processing of norms provides for satisfaction of requirements R3 and R4 enumerated in Section 2.2, in that the status of a norm can be reported on, and the arcs transitioned provide rudimentary explanations of why a

given norm has the status reported. Note that although not addressed in this paper, Section 6 describes future work addressing generation of more comprehensive explanations.

4.3 Mapping Norms to Transition Networks

We have described how transition networks capture the semantics of our norm representation, and the lifecycle of norms through their activation and expiration. In this section we formally define the mapping from an abstract norm to a transition network, grounding the more general processing of such networks discussed above.

We begin by considering the labels of arcs in transition networks. Suppose we have two normative systems $\mathcal{NS}1$ and $\mathcal{NS}2$ with norms $\mathcal{N}1$ and $\mathcal{N}2$ and their transition networks $TN_{\mathcal{N}1}$ and $TN_{\mathcal{N}2}$ respectively. The fact that $\mathcal{N}1$'s activation condition A holds, is observed and reported on by observer $O1$ (as agreed to by the agents in $\mathcal{NS}1$). Now, suppose that $\mathcal{N}2$ has the same activation condition A that is observed by $O2$ (as agreed to by the agents in $\mathcal{NS}2$). In such a situation, a monitor must be able to identify which transition network it must process. It would clearly be inappropriate if the monitor is informed by $O1$ that A holds, and then monitors for fulfillment of the norm represented by $TN_{\mathcal{N}2}$. However, if the relevant arc in $TN_{\mathcal{N}1}$ is labelled by both the activation condition A *and* $O1$, then the monitor knows that it is $TN_{\mathcal{N}1}$ that it should process. For this reason, and because monitors may monitor multiple normative systems, the labels of arcs in a norm's transition network include the observer identifiers that are uniquely entrusted by the normative system's agents to relay the truth of predicates that label each arc. A consequence of this is that it additionally enables a monitor to identify which observers to subscribe to when processing transition networks.

Now, recall that the norm components *NormActivation*, *NormCondition*, and *NormExpiration* in Section 3's general semantic model of norms, are assumed to be representable as canonical disjunctive normal form (*DNF*) clauses: $\alpha_1 \vee \ldots \vee \alpha_n$ where, for $i = 1 \ldots n$, α_i is a conjunction, $\beta_1 \wedge \ldots \wedge \beta_m$. Here, for $j = 1 \ldots m$, β_j is either a complex temporal expression, or a predicate formula that is an atomic predicate or such a predicate preceded by $\neg$. Each predicate formula is a description of a state of interest or an action description, including a message sent or received by an agent, where we assume that the action description is the single argument of the predicate *happened*.

As discussed in Section 4.1, each *DNF* representation of a norm component is explicitly agreed, by the agents in the normative system containing the norm, to *count as* the norm component it represents. The observers entrusted by these agents to observe and report on the states of interest specified in these representations are then identified for each β. In general, each temporal expression or predicate formula β is associated with a unique observer, Ob_β, which sends message M_β to the monitor, informing it that β does or does not hold.[7] In what follows, we

[7] Notice that if what is being observed is an action that is not a message, then Ob_β may instead observe for a predicate description of the postcondition of the action. Whether one includes a direct reference to the action with *happened*(...), or to a predicate description of the state of interest brought about by the action, impacts on the flexibility which with a norm

thus assume a function, *map*, which maps a predicate formula to an observer and message (or other state or action):

$$map : \beta \mapsto (Ob_\beta,\ M_\beta)$$

For the traffic warden obligation in Table 5, the conjuncts in the activation condition $\beta_1 \wedge \beta_2$ are mapped as follows (where variables are denoted by strings beginning with upper case letters):

$\beta_1 \mapsto (ob_{parking\_attendent},\ \{park(Car, yellow\_line)\})$,
$\beta_2 \mapsto (ob_{calendar_1},\ \{(2pm \leq Now \leq 3pm)\})$

We can now formally define the mapping of a norm to a transition network. Before doing so, recall that in Section 4.2.1 we illustrated how by negating $NormCondition$ we may obtain another DNF representation, where if any one of the disjuncts does not hold, then the network is transitioned across the corresponding arc to the violated state $S3$. For example, given $NormCondition$ $(\beta_1 \wedge \beta_2) \vee (\beta_3))$, we first negate $NormCondition$, and then by the standard application of De Morgan's laws obtain the equivalent conjunctive normal form (CNF) formula $(\neg\beta_1 \vee \neg\beta_2) \wedge (\neg\beta_3)$, which can be expressed in its equivalent DNF as $(\neg\beta_1 \wedge \neg\beta_3) \vee (\neg\beta_2 \wedge \neg\beta_3)$. In what follows, therefore, when we write $\neg\phi$, where ϕ is a formula in DNF, we assume that $\neg\phi$ is the equivalent CNF formula obtained by application of De Morgan's laws, and for any CNF formula ψ, we write $f_{DNF}(\psi)$ to denote the equivalent DNF representation of ψ. Also, in the following definition we will as an abuse of notation use the logical conjunction connective to denote the conjoining of tuples returned by the function *map*.

Definition 1 (**Formal mapping of norms to *transition networks***)
If ϕ be a DNF formula $(\beta 1_1 \wedge \ldots \wedge \beta 1_m) \vee \ldots \vee (\beta k_1 \wedge \ldots \wedge \beta k_n)$, then

$$TN\_Lab(\phi) = (map(\beta 1_1) \wedge \ldots \wedge map(\beta 1_m)) \vee \ldots \vee (map(\beta k_1) \wedge \ldots \wedge map(\beta k_n))$$

where $(map(\beta 1_1) \wedge \ldots \wedge map(\beta 1_m))\ , \ldots , (map(\beta k_1) \wedge \ldots \wedge map(\beta k_n))$ are individually referred to as the disjuncts in $TN\_Lab(\phi)$.

If $\mathcal{N} = (NormType,\ NormActivation,\ NormCondition,\ NormExpiration,\ NormTarget)$, then the sets of arcs in

$$TN_{\mathcal{N}} = (\{S1, S2, S3, S4, S5\},\ \mathcal{A}\overrightarrow{12}, \mathcal{A}\overrightarrow{23},\ \mathcal{A}\overrightarrow{24},\ \mathcal{A}\overrightarrow{34},\ \mathcal{A}\overrightarrow{43},\ \mathcal{A}\overrightarrow{35},\ \mathcal{A}\overrightarrow{45})$$

are defined as follows:

- $\mathcal{A}\overrightarrow{12} = \{(S1, S2)$ with label $L \mid L$ is a disjunct in $TN\_Lab(NormActivation\ \}$.
- $\mathcal{A}\overrightarrow{24} = \{(S2, S4)$ with label $L \mid L$ is a disjunct in $TN\_Lab(NormCondition)\ \}$.
- $\mathcal{A}\overrightarrow{34} = \{(S3, S4)$ with label $L \mid L$ is a disjunct in $TN\_Lab(NormCondition)\ \}$.
- $\mathcal{A}\overrightarrow{23} = \{(S2, S3)$ with label $L \mid L$ is a disjunct in $TN\_Lab(f_{DNF}(\neg NormCondition))\}$.
- $\mathcal{A}\overrightarrow{43} = \{(S4, S3)$ with label $L \mid L$ is a disjunct in $TN\_Lab(f_{DNF}(\neg NormCondition))\}$.
- $\mathcal{A}\overrightarrow{35} = \{(S3, S5)$ with label $L \mid L$ is a disjunct in $TN\_Lab(NormExpiration)\}$.
- $\mathcal{A}\overrightarrow{45} = \{(S4, S5)$ with label $L \mid L$ is a disjunct in $TN\_Lab(NormExpiration)\ \}$.

Let the norm components of Norm-Goods be defined as follows:

- $NormActivation = happened(\ (notify(S, B, G, in\_stock, T)\ )$
- $NormCondition =$
  $happened(\ send(cancel(B,S,G,T2))\ ) \vee$
  $(happened(\ send(accept(B,S,G,T3))\ ) \wedge payment\_received(S,F,G,T4)) \vee$
  $(Now \leq T + 7)$
- $NormExpiration =$
  $happened(\ send(cancel(B,S,G,T2))\ ) \vee$
  $(happened(\ send(accept(B,S,G,T3))\ ) \wedge payment\_received(S,F,G,T4)) \vee$
  $(Now > T + 7)$

Table 6 Example: **Goods Obligation Example**

We illustrate the above mapping with the Goods obligation in Table 6. The *transition network* for the norm is shown in Figure 6 (though we omit reference to the observers identified in the mapping). Notice that when defining the labels of arcs transitioning to the violated state $S3$, we first obtain

$$\neg happened(\ send(cancel(B,S,G,T2))\ ) \wedge$$
$$(\neg happened(\ send(accept(B,S,G,T3))\ ) \vee payment\_received(S,F,G,T4)) \wedge$$
$$\neg\ (Now \leq T + 7)$$

by application of De Morgan's laws, and then representing in DNF we obtain the two disjuncts labelling the two arcs from $S2$ to $S3$ and from $S4$ to $S3$.

We conclude by observing that while we have not exemplified permissions, examples of permissions will be described in the use case validation in Section 5.

4.4 Processing of *Transition Networks* by Monitors

This section describes an implementation of a monitor that receives messages from observers, and processes them so as to transition the *transition network* representations of the norms being monitored. At its core, our monitor contains a message store that is updated by received messages. When an arc is satisfied (see Definition 2 below) with respect to the contents of a message store, the monitor transitions the *transition network*. Recalling our discussion at the end of Section 3, for any norm $\mathcal{N}$ we consider its *abstract* $TN_{\mathcal{N}}$ and its *instantiated* $TN_{\mathcal{NI}}$, where $TN_{\mathcal{N}}$ is in state $S1$ and is said to be abstract because its arcs are labelled by expressions whose variables will be instantiated by concrete situations in which the norm comes into force (is activated). Hence, given $TN_{\mathcal{N}}$, when an arc a in $\mathcal{A}\overrightarrow{12}$ is satisfied, the resulting grounding of the variables in the M_βs labelling a is propagated to the variables in expressions labelling the remaining arcs in $TN_{\mathcal{N}}$, thus creating the instantiated instance $TN_{\mathcal{NI}}$, where $TN_{\mathcal{NI}}$ is then transitioned to $S2$ (corresponding to activation of the norm).

When norms are fulfilled or violated, the monitor generates notifications to the manager to take appropriate action. We illustrate this operation in Figure 7, which shows the flow of messages from the observers to the monitor's message queue, its processing and subsequent notifications to the manager.

can be fulfilled. In the latter case, the norm's targets have some flexibility in terms of the actions executed to bring about the state of interest.

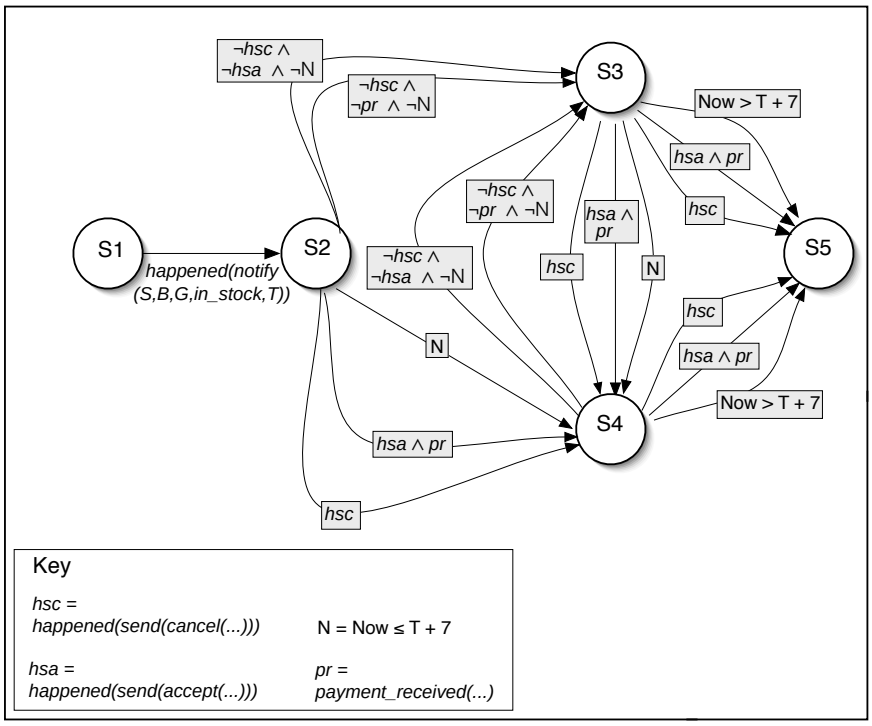

Fig. 6 *Transition network* representation of Example 1's Norm Goods obligation

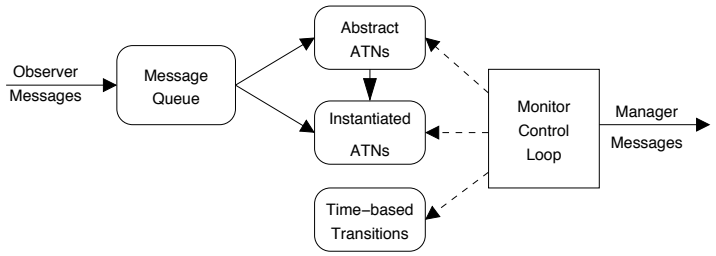

Fig. 7 Overview of the monitor control loop.

This process is more precisely illustrated in Algorithm 1, which describes the control loop used in our monitor. The algorithm makes use of the test function $satisfied(M_{St}, lab(a))$ which is defined as follows:

Definition 2 Let M_{St} be a message store and $lab(a)$ denote the label of an an arc a, where $lab(a)$ is of the form $(map(\beta_1) \wedge \ldots \wedge map(\beta_n)) = (\ (Ob_{\beta_1},\ M_{\beta_1}) \wedge \ldots \wedge (Ob_{\beta_n},\ M_{\beta_n})\ )$. Then:
$satisfied(M_{St}, lab(a))$ returns true iff for $i = 1 \ldots n$, $M_{\beta_i} \in M_{St}$, and M_{β_i} is received from Ob_{β_i}.

As long as the monitor is active, the algorithm loops. It operates by retrieving a message from the message queue, and adding it to the message store (Lines

2–4). The algorithm then checks whether any abstract norms can be instantiated (Lines 7–13). This is done by checking whether an arc from $S1$ to $S2$ is satisfied (Line 7). If so, an instantiated version of the norm is created and added to the set of instantiated norm transition networks, in state $S2$ (line 10). The remainder of the algorithm, starting at line 15, operates on instantiated norms. Lines 18–26 check whether any arc transitions from the current state can occur.[8] If so, the transition is made (Line 21), in which case Lines 29 – 42 result in the manager being notified of the transition; Line 31 informs the manager of a violation, and similarly, Lines 32 – 35 inform the manager whether a permission has started or stopped executing. Lines 37 and 40 then inform the manager of expiration and norm compliance respectively. Norm instantiation is reported in Line 11.

5 Validation: Monitoring of Contractual Clauses

In this section we report on a proof of concept implementation that has been used to validate our approach to monitoring. The work reported is more comprehensively described in [24]. Specifically, we have implemented and deployed a monitor in a prototype multi-agent system in which agents exchange messages that correspond to obliged, prohibited and permitted behaviours encoded in an electronic contract. Recent work on electronic representations and software tools for contracts [30] has highlighted a number of case studies [17], including one for aerospace logistics [22]. This involves aerospace agents — airline operators (AOs), engine manufacturers (EMs), and service sites (SSs) — whose behaviours are required to comply with (amongst others) norms governing the repair of engines and sourcing of parts for these repairs. In particular, it is commonplace for EMs, located at airports, to be under obligation to have operational engines available for the planes of a client AO. Furthermore, AOs may dictate permissions and prohibitions on the sourcing of parts (provenance restrictions) for their engines. These norms are then inherited in contracts between EMs and service sites responsible for the actual servicing and repair of engines. For instance, in order for a given *EM Rolling Royce* to fulfill its obligations and provenance restrictions for a given AO, *Rolling Royce*'s contract C with a service site *Heathhedge* stipulates a contractual obligation on *Heathhedge* to repair engines in a given time, and prohibitions and permissions on *Heathhedge* on the ordering of parts. Examples of these contractual norms, as represented in Section 3's normative model, are shown in Tables 7, 8, and 9.

Notice that the prohibition on sourcing of parts is modelled as an obligation not to source parts, and both this obligation and the permission on sourcing parts do not have expiration conditions; they remain in force indefinitely (that is, as long as *Heathhedge* remains a party to the contract).

Our proof-of-concept prototype implements the monitoring architecture in Section 4.1. Specifically, it implements:

- the normative system's participants (i.e., the contract parties) *Rolling Royce* and *Heathhedge* and the part manufacturers in the environment, as agents in the multi-agent programming language *AgentSpeak(L)* [31];

[8] Notice that starting with $j = 3$ ensures that transitions from $S4$ to $S3$ are checked before transitions from $S4$ to $S5$, thus enforcing the procedural requirement discussed in Section 4.2.3.

Algorithm 1 Monitor control loop

Require: Message queue Q_{msg}
Require: Message store M_{St}
Require: Set of abstract norm transition networks $\mathcal{X}_{Abs}$
Require: Set of instantiated norm transition networks $\mathcal{X}_{Inst}$
while Monitor is active **do**
  while Q_{msg} is not empty **do**
    Retrieve Msg from head of Q_{msg}
    Add Msg to M_{St}
    for all Abstract norm transition network A in $\mathcal{X}_{Abs}$ **do**
      for all Arcs a in $\mathcal{A}\overrightarrow{12}_A$ **do**
        if $satisfied(M_{St}, lab(a)$ **then**
          Create a norm transition network instance I from A
          Add I to $\mathcal{X}_{Inst}$
          move I to state $S2$
          Notify manager of instantiation
        end if
      end for
    end for
    for all Instantiated norm transition network I in $\mathcal{X}_{Inst}$ **do**
      transition=false
      $s = X$ where I is in state SX
      for $j = 3 \ldots 5$ such that $j \neq s$ **do**
        for all Arcs a in $\mathcal{A}\overrightarrow{sj}_I$ **do**
          if $satisfied(M_{St}, lab(a))$ **then**
            move I to state Sj
            transition=true
            break
          end if
        end for
      end for
      if !transition **then**
        continue
      else
        if I is in state $S3$ and I is an obligation **then**
          Notify manager of violation
        else if I is in state $S3$ and I is a permission **then**
          Notify manager of permission execution
        else if I is in state $S4$ and I is a permission **then**
          Notify manager that the permission is not being executed
        else if I is in state $S5$ **then**
          Notify manager of expiry
          Remove I from $\mathcal{X}_{Inst}$
        else
          Notify manager of compliance
        end if
      end if
    end for
  end while
end while

– an *AgentSpeak(L)* observer agent that is assumed to be entrusted by the contract party agents to observe messages sent and received by the contract parties; and
– an *AgentSpeak(L)* monitor agent that uses Section 4.4's algorithm to process the *transition networks* and messages relayed to the monitor by the observer.

Moreover, it includes mechanisms to deal with norms in the following way.

Table 7 Repair time Obligation

$NormType$	obligation
$NormActivation$	Order for repair of engine E received from *Rolling Royce* at time T
$NormCondition$	either engine E repaired, or it is less than 7 days after receipt of order
$NormExpiration$	engine E repaired, or it is 7 days or more after receipt of order
$NormTarget$	*Heathhedge*

Table 8 Part Sourcing Prohibition (modelled as an obligation).

$NormType$	obligation
$NormActivation$	Order for repair of engine E received from *Rolling Royce*
$NormCondition$	no order for a part P for engine E is placed to manufacturer *Cmans*
$NormExpiration$	
$NormTarget$	*Heathhedge*

Table 9 Part Sourcing Permission.

$NormType$	permission
$NormActivation$	Order for repair of engine E received from *Rolling Royce*
$NormCondition$	order for part P for engine E placed to manufacturer *Loylands*
$NormExpiration$	
$NormTarget$	*Heathhedge*

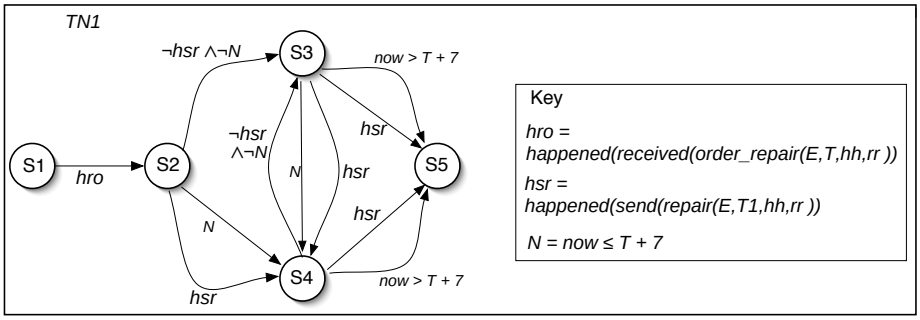

Fig. 8 *TN1* for Repair time Obligation in Table 7

– The contractual norms described above are mapped to *transition networks* labelled by messages sent and received by the above agents who are assumed to have explicitly agreed that the above messages count as the various components of the above norms. These *transition networks* (*TN1*, *TN2*, and *TN3*) are shown in Figures 8 and 9, in which 'rr' and 'hh' respectively abbreviate '*Rolling Royce*' and '*Heathhedge*', and 'cm' and 'll' respectively abbreviate '*Cmans*' and '*Loylands*'.

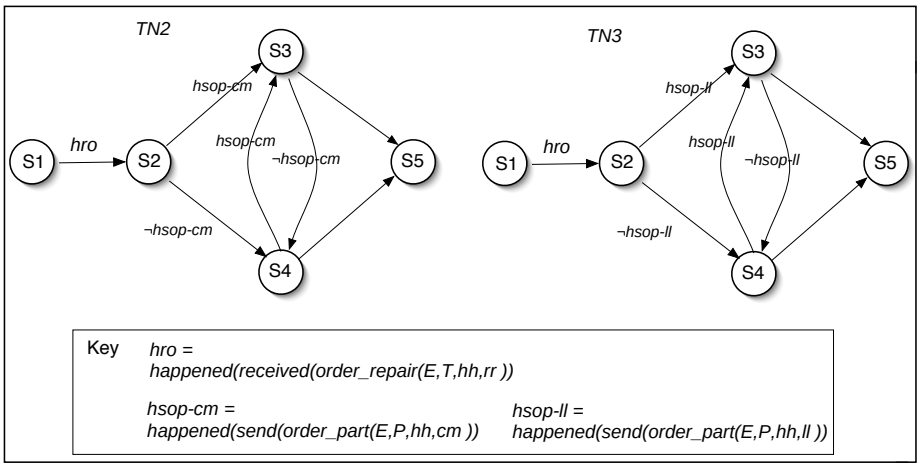

Fig. 9 *TN2* for Part Sourcing Prohibition described in Table 8, and *TN3* for Part Sourcing Permission described in Table 9.

– Based on the above processing, the monitor sends norm status reports that are displayed in a graphical user interface (see Figure 10) that is a proxy for the manager.

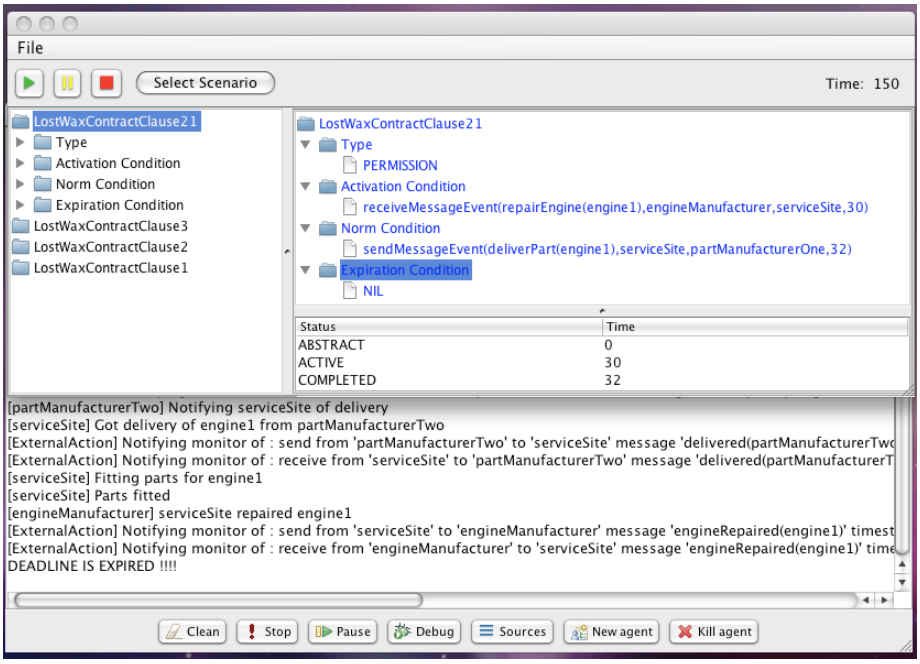

Fig. 10 Screenshot of graphical user interface.

The prototype is set up to execute a number of scenarios in which messages are exchanged between *Rolling Royce* and *Heathhedge*, and between *Heathhedge* and part manufacturers *Cmans* and *Loylands*. *All* messages exchanged are observed, and relayed to the monitor. On each time tick (every second), the monitor processes the *transition networks* together with messages received from the observer, and relays status reports for display in the graphical user interface (GUI) shown in Figure 10. For convenience, all executed scenarios interpret day long time periods as minutes, and begin with *Heathhedge*'s receipt of a message $order\_repair(engine1, 13.00,$ *Heathhedge*, *Rolling Royce*). Subsequent to the observer relaying $happened(received(order\_repair(engine1, 13.00,$ *Heathhedge*, *Rolling Royce*))) to the monitor's message store, the monitor creates instantiations of *TN1*, *TN2*, and *TN3*, each of which is transitioned to $S2$ by the monitor, with the variable E in the arc labels instantiated for $E = engine1$, and T instantiated by 13.00 in the arc labels of *TN1*.

Thus, activation of each norm is reported to the GUI (where this information can in turn be relayed to the norm's target *Heathhedge*). For each scenario, on the next time tick after activation the following apply.

1. *TN1* is transitioned to (and reported as being in) the non-violated state $S4$ given that the current time (*now*) is before 13.07.
2. *TN2* is transitioned to (and reported as being in) the non-violated state $S4$ given that the observer reports that $\neg happened(send(order\_part(engine1, P,$ *Heathhedge*, *Cmans*))) holds true.[9]
3. *TN3* is transitioned to (and reported as being in) the non-executed state $S4$ given that the observer reports that $\neg happened(send(order\_part(engine1, P,$ *Heathhedge*, *Loylands*))) holds true.

Table 10 NormMonitoringScenarios.

*Scenario*1	Repair obligation on *Heathhedge* expired and fulfilled.
*Scenario*2	Repair obligation on *Heathhedge* expired and violated.
*Scenario*3	Part sourcing prohibition violated by *Heathhedge*
*Scenario*4	Part sourcing permission executed by *Heathhedge*

Table 10 lists four implemented monitoring scenarios. In Scenario 1, the observer reports that $happened(send(repair(engine1,$ 13.04, *Heathhedge*, *Rolling Royce*))) holds true, and so *TN1* is transitioned from $S4$ to the expired state $S5$, with an accompanying report to the GUI stating that the obligation has expired having been fulfilled. Notice how this example illustrates an 'unwise' choice of observation for counting as fulfillment of the obligation, given that there is a possibility of *sanction avoidance*: *Heathhedge* may not actually have repaired the engine within seven days, but can avoid sanction by sending the message informing that the repair has been done, so that the obligation is reported as having been fulfilled.

In Scenario 2, at 13.08, $\neg(now \leq 13.00 + 7)$ holds and no message informing of repair is observed, so that *TN1* is preferentially transitioned to $S3$ *and then* $S5$, and the obligation is reported as having been violated and expired. In Scenario 3,

[9] Note that we are assuming a negation as failure interpretation of $\neg$ in the sense that $\neg happened(\ldots)$ is evaluated to true iff $happened(\ldots)$ is not evaluated as being true.

the observer reports that $happened(send(order\_part(engine1, bearing,$ *Heathhedge*, *Cmans*))) holds true, and so *TN2* is transitioned from $S4$ to $S3$, with an accompanying report to the GUI stating that the prohibition has been violated. In Scenario 4, the observer reports that $happened(send(order\_part(engine1, bearing,$ *Heathhedge*, *Loyland*))) holds true, and so *TN3* is transitioned from $S4$ to $S3$, with an accompanying report to the GUI informing that the permission has been executed.

The above prototype illustrates a number of the features of this paper's proposed framework for monitoring, with particular emphasis on the framework's modular representations of norms as *transition networks* that do not commit to specific representation of, or inter-dependencies between, norms in the normative system being monitored (requirement R5 in Section 2.3), and requirements on monitors to inform on the varying status of norms (requirements R3 and R4 in Section 2.2). Note that with regard to the latter two requirements, the monitor's explanations of the status of norms (in particular explanations as to why a norm is violated) is based on the labels of the specific arcs transitioned to the *transition networks*' corresponding state. In the following section we point to future work addressing generation of more comprehensive explanations.

6 Future Work

Providing explanations of norm violations is particularly important if managers are to assign responsibility and apply sanctions appropriately. Explanations can also help to evolve the normative specification of a system in order to prevent future violations. Thus far, our monitors provide limited explanation of violation and fulfilment, in terms of the labels of the arcs transitioned. While such explanations help to pinpoint the immediate cause of a norm's violation, they do not help to determine the overall circumstances relevant to violation, required for explaining the indirect causes leading to violation. In future work we will therefore focus on ways to recognise other observations of relevance to a violation, thus building a richer picture of the surrounding circumstances.

One obvious starting point for building richer explanations would be to utilise *all* the observations relayed to a monitor, and not only those observations that are matched to the arcs of *transition networks*. For example, in Section 5's monitoring of an electronic contract, observers are entrusted to observe *all* messages received and sent from *Heathhedge*; in particular, all messages sent to, and received from part manufacturers. The availability of these messages could provide for more comprehensive explanations. For example, *Heathhedge*'s violation of its prohibition on ordering parts from *Cmans* may have been because of the non-availability of the requested part from the permitted manufacturer *Loylands*, as indicated by *Heathhedge* sending a request for the part to *Loylands*, and *Heathhedge*'s receipt of *Loylands*'s reply denying the request given that the part is not in stock.

Given access to all observations, one can additionally enhance explanations by accounting for the fact that a single observation may be responsible for transition of multiple *transition networks* (as illustrated in Section 5). This means that violations or fulfilments of norms may have a common cause, and we can enhance our explanations of one violation by reporting the observations causing transitions in other *transition networks* with the same arc labels. For example, an engine re-

quiring repair is an activation condition for both returning the engine to working order, and for ordering parts necessary to make repairs, where each is monitored separately. If the engine is not repaired in time (so violating the first obligation) then observations relating to the ordering of parts may help explain why this was the case (e.g., because of a delay in delivery of a part from a part manufacturer). An explanation using this enhancement can thus take the form of a set of single *transition network* explanations chained together by the observations common to each *transition network*.

A monitor matches observations with the labels of *transition networks* representing norms, and so the observations received are those that are relevant only to the norms being monitored. This is often inadequate for good explanations. For example, a contract may place no obligations, prohibitions or permissions on how engine parts are supplied, and so the monitor will receive no observations regarding this supply, but it may be exactly this factor that has led to the violation of the obligation to repair an engine (i.e. something wrong with the part supply chain). We aim to improve explanations generated by the same monitoring machinery described in this paper, by adding *transition networks* representing norms not present in the contract but helpful purely for building better explanations: *explanation transition networks*. So, in Section 5's example, we would want to add an *explanation transition network* (*etr*) for monitoring the part supply chain, to ensure the monitor has records of observations relating to part supply and so can determine where part supply problems and repair violations had a common cause. An *etr* would have no qualitative difference from a *transition network* representing a norm, and would be of type permission (rather than obligation or prohibition), because it does not state what should happen, but what could happen; for example, parts could be delivered by this supplier. There would, therefore, be no notion of an *etr* itself being violated.

The focus of this paper has been on *corrective* monitoring, whereby critical states are monitored for violation of norms. *Predictive* monitoring requires representation and recognition of danger states, which are associated with agent behaviours that suggest that a norm may be in danger of violation. Future work will address how such states may be identified empirically; for example by observing and analysing violation of norms at runtime and the intermediate states that are reached prior to violation. These intermediate states can then be represented explicitly (as extra nodes) in the *transition network* representation of norms, so that during future run-time executions, observation of messages indicating transition to these states may signal preemptive action to avoid violation. To illustrate, consider that *Loylands* denied request for a part may suggest that *Heathhedge* is in danger of violating its obligation to repair an engine in 7 days, or indeed, violating its prohibition to order parts from *Cmans*. Thus, in the case of *TN1*, one could add arcs from $S2$ and $S4$ to additional nodes D_{2-3} and D_{4-3} respectively, where each arc is labelled by the message denying the part request. D_{2-3} and D_{4-3} would represent danger states, and would both be then linked by arcs to $S3$, where these latter arcs would have the same labels as $(S2, S3)$ and $(S4, S3)$, indicating violation of the obligation.

7 Related Work

Existing work in monitoring compliance with contractual obligations tends to focus on representations of the contracts tailored for the purpose, so playing the same role as our transition networks. There is much less emphasis on the possible architectures to support accurate monitoring or the requirements which inform them, with approaches tending to favour a single, independent, trusted intermediary between the contract parties, able to gather all the information required for monitoring, sometimes in conjunction with the parties themselves. However, [8], do propose an architecture for overlaying multi-agent system normative knowledge bases in which norms are represented as conditional rules. The architecture also specifies monitoring and managing components that receive notifications of events and actions, and processes these to determine the status of, and enforce, norms. However, the monitoring functionality is described in abstract terms; the focus of the work being on dynamic run-time assignment and re-assignment of roles, rights and responsibilities to agents in normative multi-agent systems.

Requirements for contract monitoring systems are briefly considered by Neal et al. [28]. Specifically, they consider non-functional requirements of a contract system with monitoring: that a contract monitoring system should be platform-neutral; integrate in a simple manner with existing systems; have accuracy of reporting in a system with distributed clocks (so no single view of time); be able to scale as more activity needs to be monitored; have security in exchange of information to give confidence to monitoring outputs. They aim to meet these requirements with a monitoring-tailored Business Contract Language (BCL), and supporting components. BCL expresses obliged activity of system components ('agents' in our terms), uses explicit time periods in which actions may/may not occur, and employs a hierarchy (ontology) of monitored actions.

A number of works explicitily consider monintoring of norms in contracts. Work such as [14, 15] has investigated reasoning over business contracts in RuleML. This work transforms a contract, expressed in RuleML, into defeasible logic, following which reasoning can be performed to identify whether normative violation occurred. While defeasible logic has been extended to cater for many different facets of contractual reasoning (for example, by incorporating deadlines [13]), it is not designed to deal with issues such as multiple repeated violations of a norm, and representing this situation is thus cumbersome at best.

With regard to our use of Augmented Transition Networks (ATNs) for representing states of norms in contracts, early work in AI and Law have also used ATNs to model and represent contracts. [12] employed ATNs for representing a kind of legal grammar of rules for "parsing" events having to do with offer and acceptance. With each new event, such as a telephone enquiry or receipt of a letter, the ATN determined the legal "state of affairs" as to whether there was a binding contract. Furthermore, during the 1990s a series of workshops explored the use of logical rules interpreting the United Nations Convention on the International Sale of Goods, to deduce the legal state of affairs as other kinds of events in a contract dispute occurrence. [38] is a good example of this work.

Others provide languages in which monitored contracts can be expressed. In work by Daskalopulu et al. [6], a contract is represented as a finite state machine (FSM), with actions in the environment (possibly initiated by agents), and the resultant changes in the status of norms causing transitions between states. They

then treat contract monitoring as the problem of determining which node of the FSM represents the current state. Their formulation of the problem assumes fallible observers of contract states, with each observer reporting back on what it believes is the current contract state. Subjective logic operators are then used to aggregate these observations, and identify the most likely contract state. Given the focus on evidence aggregation, their work attacks the problem of contract monitoring at an abstract level, and no representation of the environment, contract, or norms is suggested. Thus, states representing norm fulfilment, violation, and other changes in the status of a norm must be encoded as part of the FSM. [27] also specify a contract as sets of finite state machines, one for each party, so that the parties taking part in a contract can monitor their own activity and know what they can or must do next to ensure they meet their obligations and permissions. As with our approach, state changes in the FSMs are triggered by state change, where these changes correspond to communicative acts taking place. By placing the FSM monitoring in centralised components, through which the contract parties are required to communicate, violations of the contracted behaviour can be detected and reported.

Some approaches consider norm monitoring at a more social level, as opposed to at the level of individual interactions. For example, [20] present a norm-based model of interaction between agents operating as part of an electronic institution. They identify three different types of norms, namely institutional norms, constitutional norms, and operational norms. Institutional norms govern the behaviour of all agents within the institution, and represent the commitments of agents towards the institution as a whole. Institutional norms thus represent the "social contract" to which all agents operating within the institution are bound. Constitutional norms are used to form virtual organisations, and regulate the virtual organisation's behaviour. They apply only to members of the organisation. Operational norms specify contracts between agents, and are narrower in scope than those of constitutional norms. Monitoring takes place through rules represented as an institutional norm. Institutional norms are presented, allowing agents to determine whether an obligation is fulfilled or violated, and sanctioning norms (that is, contrary to duty obligations) are similarly defined. The focus of the paper is on the hierarchy of norms and the structure of the institutions, and monitoring is thus presented as an example of the power of their framework, without going into its applications or requirements in depth.

Xu and Jeusfeld [36] provides a formalisation of commitments which allows their fulfilment to be monitored by the agent who has made those commitments and, in particular, for obliged actions to be triggered. The commitments are modelled as sequences of actions with deadlines, and dependencies between them encoded in temporal logic. By treating the dependencies as guard conditions, a monitoring component can match actions against these conditions and notify the agent of subsequent permissions and obligations. Taken to a more social level, Xu et al. [37] extend their work to provide means to determine the blame in a system of multiple agents bound by interconnected commitments. This allows them to detect whether the cause of one agent having violated their obligation was that another agent on which they depended had violated their obligation. Our own consideration of mitigating circumstances for violations such as these, and how parties may handle them, are discussed elsewhere [25].

In [33], van der Torre and Tan describe DIO(DE)^2, a formalism which provides diagnostic and decision theoretic reasoning over norms. The former allows for the identification of violation in historic contexts, while the latter predicts the effects of actions on a system. DIO(DE)^2 thus provides a form of monitoring, enabling the identification of norm violations and norm compliance. However, unlike the work presented here, their approach is theoretical in nature, with little consideration given to practical issues, or for the modelling of complex real world norms. Furthermore, no attention is given to issues such as communication between the monitor and interested parties within the system.

The approach proposed by Fagundes *et al.* [10] considers norm enforcement within stochastic environments and in which enforcement has a cost, and analyses the tradeoffs between enforcement intensity and the cost of this enforcement. Such an approach can be seen as an extension of the approach in this work that adds stochasticity and costs to the environment. Nevertheless, the norm representation used in the mechanism is much simpler than our work. Integrating such approaches provides an interesting avenue for future research. Under a similar stochastic approach, Oh *et al.* [29] use a simple norm representation and plan recognition algorithms to compute the probability of norm violations for humans planning for multiple objectives. The proposed Prognostic Normative Reasoning (PNR) approach can then issue warnings and guide human users towards normatively-compliant plans.

Recently, Alechina *et. al.* [3] examine how norms can be monitored when the monitors have imperfect observational capabilities. Their focus was on optimally modifying the norms themselves in such a way that the maximal number of violations can be detected. This work is in effect the dual of [4], which describes how discrete monitors can be combined to monitor norms that cannot be observed by individual monitors. Like our work, both of these strands of research concentrate on corrective monitoring. However, they utilise a much simpler norm lifecycle than we describe in the current work, and cannot easily be extended to perform predictive monitoring. More recently, Alechina *et. al.* [2] propose an LTL-based norm formalism and examine the practical limits of enforcing such norms given limited computational power and bounded lookahead capabilities. Unlike our work, the enforcement mechanism described deals with regimented norms and uses a guard mechanism that restricts possible actions in order to avert violations in future states. However, a mechanism very similar to the one used to decide which actions to restrict can be used together with our monitoring mechanism to monitor LTL-based norms and apply sanctions rather than directly prevent violations.

While there are many complementary ideas in the existing work on electronic monitoring, which could extend our approach in fruitful directions, there are unique characteristics to our approach which we argue are necessary in monitoring the electronic activity fulfilling norms of the complexity found in business cases. In particular, most of the above approaches to monitoring do not distinguish between the data structures used for monitoring and the representations of the norms themselves. A consequence of this is that the norms (for example, in contracts) must be expressed in terms of observable events, often agent actions. In comparison, we allow norms to state declarative achievement or maintenance goals, which can be used by the agent to which the norm applies to plan its course of action, and then map these structures to a separate representation for monitoring: the *transition networks*. Another important distinction follows from this:

we explicitly consider the common case where a norm concerns multiple instances of an obligation (for example, for every order placed, goods should be delivered), and can handle these instances happening in parallel in a scalable way, through multiple independently monitored transition networks.

8 Conclusions

In this paper we have presented a framework for monitoring of electronic business systems in which the behaviours of business parties are monitored to ensure that prior commitments are adhered to, thus ensuring that business transactions can take place in a secure and committed context. We have situated the study of how to monitor fulfilment or violation of commitments in electronic business transactions, by modelling such commitments as norms that are explicitly encoded in multi-agent systems. This is because, as in electronic business transactions, the agents to whom the norms apply are assumed to be self-interested, autonomous, problem-solving entities working together to achieve some overarching objective, while satisfying their own individual needs.

In the first part of the paper we enumerated a set of requirements that we argue should be met by any general and reusable framework for monitoring of normative multi-agent systems. The requirements are primarily designed so as to ensure that:

1. some measure of assurance can be given that sanctions applied in case of violation will be applied only as and when appropriate, so encouraging participation of agents in normative systems;
2. the framework is applicable to a wide variety of types of norms of varying degrees of complexity; and
3. the framework is to some degree normative system neutral in making as few commitments as possible to the specifics of the normative system being monitored.

In the second part of the paper we described a framework for monitoring designed to meet the aforementioned requirements. In order to meet the latter two general requirements, the framework describes the deployment of monitors processing transition network representations of achievement *and* maintenance norms (that may apply to groups of agents) in the normative system being monitored, where such representations assume only that the system's norms conform to a general and widely applicable semantic model of norms.

The transition network representations are labelled by states that the system's participating agents explicitly agree to as counting as the various states that any given norm can be in, and the monitors' processing of these network representations involves the matching of these labels with observations relayed by observers who are entrusted by the participating agents to accurately observe and report these observations. These features, together with the fact that limited explanations of the various statuses of norms can be generated (so ensuring appropriate assignment of responsibility in case of violation), provide for satisfaction of the first requirement above.

In the third part of the paper we described an implementation of a monitor's processing of transition network representations of norms together with observa-

tions, and validated the utility and feasibilty of our approach by prototyping the monitoring of agents interacting in the context of an electronic business contract.

In conclusion we believe that the requirements we have set out for a general and reusable framework for monitoring are comprehensive, and, as demonstrated through our implementation, realisable.

References

1. Marco Alberti, Marco Gavanelli, Evelina Lamma, Federico Chesani, Paola Mello, and Paolo Torroni. Compliance Verification of Agent Interaction: A Logic-Based Software Tool. *Applied Artificial Intelligence*, 20(2-4):133–157, 2006.
2. Natasha Alechina, Nils Bulling, Mehdi Dastani, and Brian Logan. Practical run-time norm enforcement with bounded lookahead. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 443–451, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.
3. Natasha Alechina, Mehdi Dastani, and Brian Logan. Norm approximation for imperfect monitors. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 117–124. IFAAMAS/ACM, 2014.
4. Nils Bulling, Mehdi Dastani, and Max Knobbout. Monitoring norm violations in multi-agent systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 491–498, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.
5. R. Conte, R. Falcone, and G. Sartor. Agents and norms: How to fill the gap? *Artificial Intelligence and Law*, 7:1–5, 1999.
6. Aspassia Daskalopulu, Theo Dimitrakos, and Tom Maibaum. Evidence-Based Electronic Contract Performance Monitoring. *Group Decision and Negotiation*, 11(6):469–485, November 2002.
7. M. Dastani, D. Grossi, John-Jules Ch. Meyer, and Nick Tinnemeier. Normative multi-agent programs and their logics. In *Proc. Workshop on Knowledge Representation for Agents and Multi-Agent Systems (KRAMAS'08)*, pages 236–243, 2008.
8. Farnaz Derakhshan, Trevor Bench-Capon, and Peter McBurney. Dynamic assignment of roles, rights and responsibilities in normative multiagent systems. *Journal of Logic and Computation*, 23:355–372, 2011.
9. M. Esteva, B. Rosell, J. A. Rodrguez-aguilar, and J. Ll. Arcos. Ameli: An agent-based middleware for electronic institutions. In *3rd Int. Joint Conference on Autonomous Agents and Multi-agent Systems*, pages 236–243, 2004.
10. Moser Silva Fagundes, Sascha Ossowski, and Felipe Meneguzzi. Imperfect norm enforcement in stochastic environments: An analysis of efficiency and cost tradeoffs. In Ana L.C. Bazzan and Karim Pichara, editors, *Advances in Artificial Intelligence – IBERAMIA 2014*, volume 8864 of *Lecture Notes in Computer Science*, pages 523–535. Springer International Publishing, 2014.
11. Andrew D. H. Farrell, Marek Sergot, Mathias Salle, and Claudio Bartolini. Using the event calculus for tracking the normative state of contracts. *International Journal of Cooperative Information Systems*, 4(2–3):99–129, 2005.
12. A. Gardner. *An Artificial Intelligence Approach to Legal Reasoning*. MIT Press, 1987.
13. G. Governatori, Joris Hulstijn, R. Riveret, and A. Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Proceedings of AI-2007*, volume 4830 of *Lecture Notes in Artificial Intelligence*, pages 486–496, 2007.
14. Guido Governatori. Representing business contracts in ruleml. *International Journal of Cooperative Information Systems*, 14(2–3):181–216, 2005.
15. Guido Governatori and Antonino Rotolo. Modelling contracts using *RuleML*. In *Proceedings of Jurix 2004*, Amsterdam, The Netherlands, 2004.
16. D. Grossi. *Designing Invisible Handcufffs*. PhD thesis, Utrecht University, SIKS, 2007.
17. M. Jakob, M. Pchouek, J. Chabera, S. Miles, M. Luck, N. Oren, M. Kollingbaum, C. Holt, J. Vazquez, P. Storms, and M. Dehn. Case studies for contract-based systems. In *Proc. 7th Int. Joint Conference on Autonomous Agents and Multiagent Systems*, pages 55–62, 2008.

18. Andrew J. I. Jones and Marek Sergot. On the characterisation of law and computer systems: The normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*, pages 275–307. John Wiley and Sons, 1993.
19. Martin Kollingbaum. *Norm-governed Practical Reasoning Agents*. PhD thesis, University of Aberdeen, 2005.
20. Henrique Lopes Cardoso and Eugenio Oliveira. Electronic institutions for B2B: dynamic normative environments. *Artificial Intelligence and Law*, 16:107–128, 2008.
21. F. Lopez Y Lopez, M. Luck, and M. d'Inverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12(2-3):227–250, 2006.
22. F. R. Meneguzzi, S. Miles, M. Luck, C. Holt, M. Smith, N. Oren, N. Faci, M. Kollingbaum, and S. Modgil. Electronic contracting in aircraft aftercare: A case study. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, Industry and Applications Track*, 2008.
23. Felipe Meneguzzi, Sanjay Modgil, Nir Oren, Simon Miles, Michael Luck, Nora Faci, Camden Holt, and Malcolm Smith. Monitoring and explanation of contract execution: A case study in the aerospace domain. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, pages 77–84, 2009.
24. F.R. Meneguzzi, S. Modgil, N. Oren, S. Miles, M. Luck, and N. Faci. Applying electronic contracting to the aerospace aftercare domain. *Engineering Applications of Artificial Intelligence*, 25(7):1471–1487, 2012.
25. Simon Miles, Paul Groth, and Michael Luck. Handling Mitigating Circumstances for Electronic Contracts. In *Proceedings of the AISB 2008 Symposium on Behaviour Regulation in Multi-agent Systems*, pages 37–42. The Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2008.
26. Sanjay Modgil, Noura Faci, Felipe Rech Meneguzzi, Nir Oren, Simon Miles, and Michael Luck. A framework for monitoring agent-based normative systems. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, pages 153–160, 2009.
27. Carlos Molina-Jimenez, Santosh Shrivastava, Ellis Solaiman, and John Warne. Run-time monitoring and enforcement of electronic contracts. *Electronic Commerce Research and Applications*, 3(2):108–125, 2004.
28. S. Neal, J. Cole, P.F. Linington, Z. Milosevic, S. Gibson, and S. Kulkarni. Identifying requirements for Business Contract Language: a monitoring perspective. In *Proceedings of the Seventh IEEE International Enterprise Distributed Object Computing Conference*, pages 50–61. IEEE Comput. Soc, 2003.
29. Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy J. Norman. Prognostic normative reasoning. *Engineering Applications of Artificial Intelligence*, 26(2):863 – 872, 2013.
30. N. Oren, S. Panagiotidi, J. Vazquez-Salceda, S. Modgil, M. Luck, and S. Miles. Towards a formalisation of electronic contracting environments. In *Proc. Coordination, Organization, Institutions and Norms in Agent Systems (COIN 2008)*, pages 156–171, 2008.
31. Anand S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In Walter Van de Velde and John W. Perram, editors, *Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, volume 1038 of *LNCS*, pages 42–55. Springer, 1996.
32. J. R. Searle. *The Construction of Social Reality*. Free Press, 1997.
33. Leendert van der Torre and Yao-Hua Tan. Diagnosis and decision making in normative reasoning. *Artificial Intelligence and Law*, 7:51–67, 1999.
34. G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.
35. W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, 1970.
36. L Xu and M A Jeusfeld. Pro-active monitoring of electronic contracts. In *Proceedings of the 15th Conference On Advanced Information Systems Engineering*, volume 2681 of *Lecture Notes of Computer Science*, pages 584–600. Springer-Verlag, 2003.
37. Lai Xu, Manfred A. Jeusfeld, and Paul W. P. J. Grefen. Detection tests for identifying violators of multi-party contracts. *ACM SIGecom Exchanges*, 5(3):19–28, April 2005.
38. H. Yoshino. Logical structure of contract law system for constructing a knowledge base of the united nations convention on contracts for the international sale of goods. *Advanced Computational Intelligence*, 2:2–11, 1998.