Towards a Monitoring Framework for Agent-Based Contract Systems

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

Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK
noura.faci@kcl.ac.uk

Abstract. The behaviours of autonomous agents may deviate from those 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, and may be encoded in electronic contracts in order to specify the obliged, permitted and prohibited behaviours of agents that are signatories to such contracts. Enactment and management of electronic contracts thus enables the use of regulatory mechanisms to ensure that agent behaviours comply with the encoded norms. To facilitate such mechanisms requires monitoring in order to detect and explain violation of norms. In this paper we propose a framework for monitoring that is to be implemented and integrated into a suite of contract enactment and management tools. The framework adopts a non-intrusive approach to monitoring, whereby the states of a contract with respect to its contained norms can be inferred on the basis of messages exchanged. Specifically, the framework deploys agents that observe messages sent between contract signatories, where these messages correspond to agent behaviours and therefore indicate whether norms are, or are in danger of, being violated.

1 Introduction

Interactions in systems composed of heterogeneous and self-interested agents are inherently unreliable, requiring some form of societal control to bind these interactions. The introduction of norms has been proposed to address this need in such systems [8, 1], allowing for open societies of autonomous agents that are, nevertheless, regulated to some degree. Such norms are usually specified using deontic concepts, including the notions of obligations, permissions and prohibitions that govern or direct agent behaviour in multi-agent systems. By incorporating sets of these norms into a formal document representation, it is possible to define electronic contracts, which mirror the the paper versions exchanged between businesses today, and offer the possibility of dynamic, runtime enforcement of *contract party* agent behaviours to ensure compliance with norms.

This work is situated in the context of the CONTRACT project[1] that aims to develop frameworks, components and tools that make it possible to model, build, verify and monitor distributed electronic business systems on the basis of dynamically generated, cross-organisational contracts which underpin formal descriptions of the expected

[1] www.ist-contract.org

behaviours of individual agents and the system as a whole. In this paper we report on ongoing development of the project's framework for *monitoring* electronic contracts.

The fact that agents are to varying degrees autonomous means that their behaviours may deviate from those prescribed by norms; that is, agents may violate norms. Thus there are requirements for monitoring norm violations during run-time contract-based governance of agent behaviours [14]. Monitoring has also been extensively studied in a Web Services context (e.g., [6, 12, 13]). However, monitoring of contracts specified as Service-Level Agreements (SLA), focuses on quality of service metrics rather than on the behaviours of contractual entities. We adopt the latter perspective in this paper; one that allows for planning-oriented detection and analysis of norm violation by agents. Two types of monitoring have been adopted in multi-agent systems:

1. In *corrective monitoring* (e.g., [5]) violations are detected as they occur, and the results of analysis of these violations are used to instigate corrective measures. Such measures might include the imposing of punishments on those contract parties that violate norms, thus motivating future compliance (as well as possibly compensating contract parties injured by non-compliance). These punishments may be manifest in the form of less favourable terms offered to the violating parties when renegotiating contracts. Furthermore, the very fact that agents are being monitored, and thus the *threat* of punishment, may itself motivate compliance.
2. In *predictive monitoring* (e.g. [18]) norm violations are predicted and actions are specified to avoid violation. The obvious advantage is that the business process can continue uninterrupted by remedial measures.

Our focus in this paper is on corrective monitoring based on run-time observation of agent behaviours. However, we also discuss how the monitoring of agent behaviours can be used for predictive purposes.

Approaches to monitoring of agent behaviours can also be distinguished according to whether they adopt an *intrusive* or *overhearing* approach. In the intrusive approach (e.g., [2, 4, 16, 9]), the mental states of agents are assumed to be available for inspection. Agents communicate their states to monitors that subsequently interpret the behaviours of agents. Intrusive approaches thus make the design of agent-based systems more complex, and rely heavily on the compliance of agents to communicate the required data. We therefore adopt the *overhearing* approach [5] in which messages exchanged among agents are observed, and behaviours are inferred from these messages.

The paper is organised as follows. Section 2 briefly describes the CONTRACT project's representation of electronic contracts, their contained norms, and the architecture for enactment and managing of contracts. Sections 3, 4 and 5 then describe the primary contribution of this paper: we propose a multi-agent framework for non-intrusive monitoring of electronic contracts, whereby the *states* of a contract with respect to its contained norms can be inferred on the basis of messages exchanged. In this way one can recognise whether a norm is currently in force, in danger of being violated, and whether a norm is in fact violated or complied with. We believe that our framework is the first to adopt such an approach to monitoring of electronic contracts.

Section 3 focuses on the entities and associated information flows that comprise the monitoring aspects of the architecture, while Section 4 then characterises the behaviour of the architecture's agents in terms of the messages these agents receive and

send. Section 5 then describes the mapping from the norms expressed in a contract to a representation suitable for monitoring, and how violations are detected based on the processing of these representations and the observed exchange of messages between contract party agents. Section 6 looks forward to future work; in particular, work on generation of explanations for norm violations, and representation of danger states that indicate an increased likelihood of norm violation and thus enable predictive monitoring. Finally, Section 7 concludes and discusses related work.

2 Norms and Architecture

The CONTRACT project encodes contracts as XML documents consisting of normative clauses that are essentially declarative specifications of agent behaviours. Associated with these documents are ontologies that describe and define background concepts and terms. These contract documents and associated ontologies are more fully described in [15], but the XML encoding of normative clauses is the primary input to the monitoring architecture. Here, we briefly review our underlying model of norms that ground specification of a contract's normative clauses.

Norms can be classified into *obligations* (what should be done), *prohibitions* (what should not be done) and *permissions* (what is allowed to be done). Norms affect *target* agents that agree to abide by the norms contained in a contract, which apply under certain circumstances. Note that these circumstances may occur multiple times during a contract's lifetime. For example, an obligation to keep a fire door shut (the obligation's goal state or *condition*) takes force whenever the door is opened. Whenever such triggering (*activating*) circumstances arise, an instantiated version of the norm parameterised by those circumstances begins to take effect on the target agent's behaviour.

More formally, norms are tuples of the form:

$$(NormType, NormActivation, NormCondition, NormExpiration, NormTarget)$$

where $NormType \in \{$obligation, permission, prohibition$\}$, $NormActivation$ denotes the conditions under which the norm is activated (triggered), and $NormCondition$ denotes the goal or state that:

- must be brought about by the $NormTarget$ in the case of an obligation;
- may be brought about by the $NormTarget$ in the case of a permission; or
- must not be brought about by the $NormTarget$ in the case of a prohibition.

Finally $NormExpiration$ denotes the conditions under which the norm is no longer in force.

Example 1. Consider the following obligation $\mathbf{Ob}_{Del}$ on an agent AgX to deliver $GoodsZ$ to agent AgY within one week of receiving the order from AgY:

- $NormType$ = obligation
- $NormActivation$ = $order\_placed(AgY, AgX, GoodsZ)$ denoting that AgY has placed an order to AgX for $GoodsZ$

- $NormCondition$ = $deliver(AgX, AgY, GoodsZ, 1\text{ week})$ denoting that AgX has delivers $GoodsZ$ to AgY within 1 week
- $NormExpiration$ = $delivered(AgX, AgY, GoodsZ, 1\text{ week})$ denoting that AgX has delivered $GoodsZ$ to AgY in 1 week
- $NormTarget$ = AgX

Consider also the permission $\mathbf{Per}_{Del}$ and prohibition $\mathbf{Pro}_{Del}$ that are defined in the same way as $\mathbf{Ob}_{Del}$, except that in the former case $NormType$ = permission, and in the latter case $NormType$ = prohibition. Thus:

- $\mathbf{Per}_{Del}$ permits agent AgX to deliver $GoodsZ$ to agent AgY within one week of receiving the order from AgY
- $\mathbf{Pro}_{Del}$ prohibits agent AgX from delivering $GoodsZ$ to agent AgY within one week of receiving the order from AgY [2]. ■

An administrative architecture [11] has also been defined, consisting of a set of service-oriented middleware components and design patterns to support management of electronic contracts. The architecture can be seen as a combination of the following stages, which are applied to an electronic contracting application as a methodological process. First, off-line verification mechanisms check whether the contracts to be enacted obey certain properties, such as being consistent, or achievable given the possible states the world can reach. The architecture also provides for definition of application specific processes suitable for administration of the electronic contracts through their lifetimes, including enactment, updating, termination, renewal, and so on. Such processes may also include observation of the system, so that the contract can be enforced or otherwise effectively managed. Once suitable application processes are identified, we can specify the roles that agents play within them, and the components that agents can utilise to allow them to manage the contracts. In the following section we give more detail on the agents, components and processes relevant to monitoring.

3 Overview of Monitoring

In this section, we introduce a novel approach to monitoring for on-line detection of contract/norm violations in contract-based systems. The approach describes observation of communications between contract parties, and performs matching of the observed communications against augmented transition networks (*ATN*s)[17] which are essentially directed labelled graphs consisting of nodes and arcs. These *ATN*s characterise acceptable, prohibited, and obliged behaviours. The contract parties are treated as black boxes and their internal state transitions are invisible to the Monitoring components.

As discussed in Section 1, our approach to monitoring is based on observation of messages received and sent by agents that are signatories to a contract (the contract

[2] The example prohibition is primarily illustrative; it is admittedly somewhat odd to have a prohibition on delivery of goods within a certain time period. We model such a prohibition given the convenient representational match with the permission and obligation.

parties). This requires that each normative clause in a contract is mapped to a representation whereby:

- the state (of the world) described by a norm's $NormActivation$ can be recognised as being brought about, on the basis of messages exchanged; and
- the state (of the world) described by a norm's $NormCondition$ as being obliged to, permitted to, or prohibited from, being brought about, can be recognised on the basis of messages exchanged.

For example, consider the obligation $\mathbf{Ob}_{Del}$ in Example 1. The obligation is activated when

$$order\_placed(AgY, AgX, GoodsZ)$$

holds, and this is recognised as being the case when the message

$$order(AgY, AgX, GoodsZ)$$

has been sent by AgY to AgX. Observation of this sent message indicates that the contract is in a *critical state* with respect to $\mathbf{Ob}_{Del}$.

The obligation is fulfilled when $\mathbf{Ob}_{Del}$'s

$$NormCondition = deliver(AgX, AgY, GoodsZ, 1 \text{ week})$$

holds. This is recognised as being the case when the message

$$notify\_delivery(AgX, AgY, GoodsZ)$$

is observed as having been sent by AgX to AgY within one week of receipt of the above $order$ message. If the $notify\_delivery$ message is not observed as having been sent within one week, then $\mathbf{Ob}_{Del}$ is deemed to be violated.

In Section 5 we further describe a framework for mapping a contract's normative clauses to *augmented transition networks* (*ATN*s) [17], which constitute the *monitoring representation*, and which associate the $NormActivation$ and $NormCondition$ constituents of norms with messages exchanged.

Figure 1 provides an overview of the monitoring architecture. The architecture has been designed based on the assumption that monitoring parties are external to the contract itself. This means that Monitors can be flexibly deployed for any contracts, provided that appropriate *ATN* representations of contracts are available for input to Monitors, and that Monitors can operate asynchronously from the execution of the contract itself. These design assumptions ensure that the system's performance and monitoring performance are independent of each other. Furthermore, failure of one will not adversely affect the other (for example if one monitor fails then another monitor can be deployed without interrupting execution of the contract).

In the architecture, a *Mapper* maps a contract (obtained from a *Contract Store*) to *ATN*s for input to the *Monitor*. This input is provided off-line. During the run-time enactment of a contract it is the actual messages exchanged that are matched by the Monitor with the *ATN*s in order to detect norm violations.

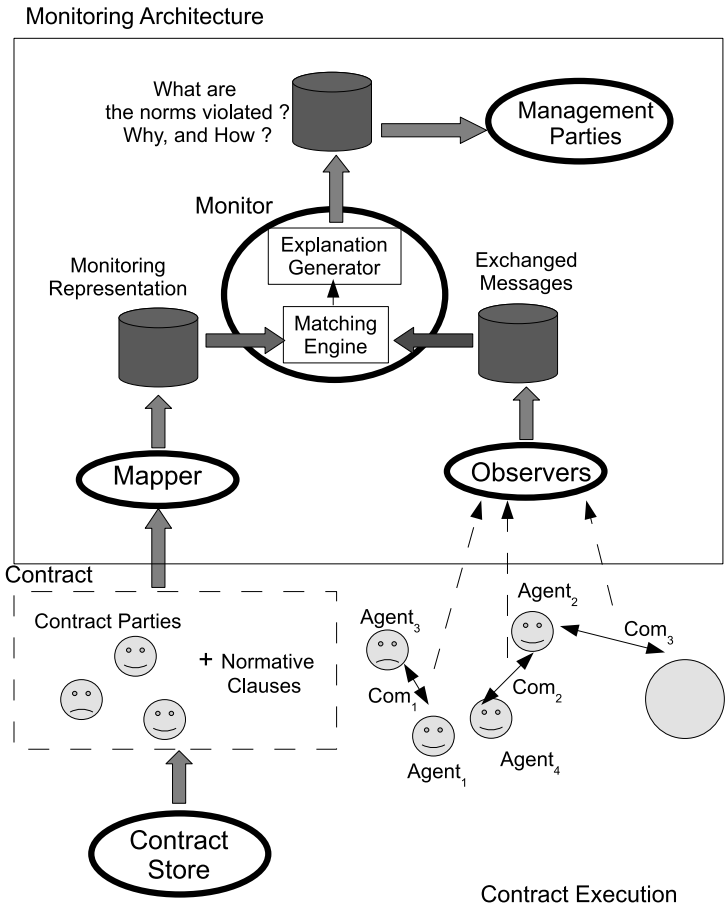

Fig. 1. Monitoring Architecture (ovals denote agents and cylinders denote data stores)

Notice that all messages exchanged between contract party agents, and between contract party agents and the *Environment* (communicative entities that are not contract party agents) must be observable by *Observers*. For effective monitoring, this requirement is mandatory, and can be enforced by enabling Observer interception of all communicative interactions. This is illustrated in Figure 1 in which the Observer *probes* each of the communication channels between agents, and between agents and the environment. These communication channels are conceptual entities; in practice, the probing may be implemented by associating the Observer with the middleware communication interfaces of each agent. Intuitively, if every normative clause is mapped to messages exchanged by entities, and all such messages are observable, then this provides some measure of guarantee that every norm violation can be monitored.

Finally, the Monitor processes each norm violation in order to provide an explanation of the violation that is made use of by *Management* party agents in order to, for

example, impose punishments in the case of corrective monitoring, or instigate preemptive action in the case of predictive monitoring. In Section 6 we look forward to future work addressing explanation generation. The following section then characterises the behaviour of the agents in Figure 1 in terms of their interfaces with other agents.

4 Agent Behaviours in the Monitoring Architecture

The monitoring architecture is intended to be integrated into a wide range of applications, and deployed in varying ways. To ease this integration process, we follow a service-oriented approach by defining the form of messages sent from and received by the monitoring components (i.e. their *interfaces*). Publishing an interface allows technology-specific agents to be implemented such that they use the correct format of messages to communicate, regardless of their internal architectures. In this section, we define the interfaces for the components introduced in the previous section. For each component, we define the form of each message type it sends, the parameters the message provides, and the role of the agent expected to receive it. Note that in what follows, messages will contain the unique names *monitor-id*, *observer-id*,. . . of agents and other components of the monitoring architecture.

4.1 Monitor

The Monitor is required to report violations of active contracts (corrective monitoring) and issue warnings when there is a risk of violation (predictive monitoring). These behaviours are specified by messages sent from the Monitor to Observers and Management parties. These messages are of the following form:

- **Subscribe(***monitor-id, observer-id, contract-id, timeInterval/eventExp***)**: The Monitor agent, named *monitor-id*, subscribes to the Observer, named *observer-id* with respect to a given contract, named *contract-id*, in order to receive observed data at intervals corresponding to period, *timeInterval*, or when a condition *eventExp* is true.
- **Cancel(***monitor-id, observer-id, contract-id***)** : The Monitor agent, *monitor-id*, cancels its subscription to the Observer, *observer-id* for *contract-id*.
- **Inform(***monitor-id, manager-id, contract-id, norm, explanation, violator(s)***)** : The Monitor agent, *monitor-id*, informs *manager-id* of a violation of a *norm* by *violator(s)* in *contract-id*, because of *explanation*.
- **Inform(***monitor-id, manager-id, contract-id, norm, explanation, violator(s), timeInterval***)**: The Monitor agent, *monitor-id*, informs *manager-id* of the existence of a danger state in which, after a period of time, *timeInterval*, after the sending of the inform message, *violator(s)* may violate *norm* in *contract-id*, because of *explanation*.

4.2 Observer

The Observer collects data (observes messages). This requires that the Observer subscribes to communication channels between agents, and between agents and the environment.

– **Subscribe(***observer-id, communication_channel-id***)** : Observer, *observer-id*, subscribes to communication channel *communication_channel-id*

Messages are relayed from the Observer to the Monitor:

– **Notify(***observer-id, monitor-id, contract-id, contract messages***)** : Observer, *observer-id*, notifies Monitor, *monitor-id*, of *contract messages* exchanged, sent and received by contract parties in *contract-id*.

4.3 Mapper

The Mapper maps the representation of the contract in a contract store to the *ATN* monitoring representation for input to the Monitor. This behaviour is specified by messages sent from the Mapper to the Contract store and Monitor:

– **Subscribe(***mapper-id*, *contractStore-id***)**: *mapper-id* subscribes to *contractStore-id*.
– **Notify(***mapper-id, monitor-id, contract-id, contractParty-id, transition-structure***)**: *mapper-id* notifies *monitor-id* with the mapped *ATN* representation, *transition-structure*, of a new active contract, *contract-id*, where this *ATN* specifies *contract messages* (see above) associated with transitions between states, as described in Section 5.

4.4 Contract Store

The Contract Store provides the Mapper with Contracts, as specified by messages sent from the Contract Store to the Mapper. These messages are of the form:

– **Inform(***contractStore-id, mapper-id, activeContract-id, norm_clauses***)**:
contractStore-id provides *mapper-id* with a new contract, *activeContract-id*, and its contained normative content, *norm_clauses*, which is to be mapped to the *ATN*s.

4.5 Manager

The Manager receives the results of monitoring from the Monitor, as specified by messages sent from the Manager to the Monitor:

– **Subscribe(***manager-id, monitor-id, contract-id***)** : *manager-id* subscribes to a Monitor, *monitor-id*, for a contract, *contract-id*.
– **Cancel(***manager-id, monitor-id, contract-id***)** : *manager-id* cancels its subscription to *monitor-id* for *contract-id*.

5 Contract Monitoring: Representation and Interpretation

This section describes the mapping of normative clauses in a contract to its monitoring representation – Augmented Transition Networks (*ATN*s) – such that the normative clauses map to messages exchanged between contract parties. It is these messages that are observed in order to determine when a contract is in a critical state with respect to a given normative clause, and whether contract parties comply with the normative clause.

5.1 Mapping Norms to Augmented Transition Networks

*ATN*s were originally developed for natural language processing, and are recursive in the sense that *ATN*s can themselves label arcs. In the *ATN* representation of normative clauses, nodes correspond to states of the contract in which norms are activated and norms may or may not be violated. Transitions between nodes are labelled by messages sent and received by contract party agents. Intuitively, the messages correspond to actions executed by agents, where these actions in turn bring about states of affairs in which norms are activated, and states of affairs in which norms may or may not be violated.

We have defined a general framework for mapping that takes as input the XML encoding of a contract and its associated OWL (www.w3.org/TR/owl-ref/) encoded domain and action ontologies [15]. Domain ontologies define the predicates used in the description of states, and the action ontologies describe the actions executed by contract party agents and interacting agents in the environment, where these action ontologies include the pre and post-conditions of the actions that are in turn described by predicates in the domain ontology.

Given norm ($NormType$, $NormActivation$, $NormCondition$, $NormExpiration$, $NormTarget$), then for N = $NormActivation$ or N = $NormCondition$, the *actors* and *actions* associated with N are identified, and respectively denoted by *actors*(N) and *actions*(N). Currently, this process of identification is not automated, and involves selection of actions in the action ontology whose post-conditions (defined in the domain ontology) match the $NormActivation$ and $NormCondition$.

A mapping is then defined that takes as input N, *actors*(N) and *actions*(N), and returns a set of messages $\mathcal{M}_N$ and synchronisation conditions $Synch_N$ on $\mathcal{M}_N$:

$$message\_map(N, \textit{actors}(N), \textit{actions}(N)) \mapsto (\mathcal{M}_N, Synch_N)$$

Intuitively, messages $\mathcal{M}_N$ are those exchanged between *actors*(N). The synchronisation conditions $Synch_N$ describe temporal relations on these messages such that if the messages are observed as specified by these temporal relations, then one can infer that *actors*(N) have executed *actions*(N) in order to bring about N. Note that in what follows we will focus on the contents of messages and will not commit to a specific agent communication language.

Example 2. Recall $\mathbf{Ob}_{Del}$ in Example 1:

- $NormType$ = obligation
- $NormActivation$ = $order\_placed(AgY, AgX, GoodsZ)$
- $NormCondition$ = $deliver(AgX, AgY, GoodsZ, 1\text{ week})$
- $NormExpiration$ = $delivered(AgX, AgY, GoodsZ, 1\text{ week})$
- $NormTarget$ = AgY

For $NormActivation$, the *actors* and *actions* are as follows:

actors($order\_placed(AgY, AgX, GoodsZ)$) = $\{AgY, AgX\}$
actions($order\_placed(AgY, AgX, GoodsZ)$) = $\{order(AgY, AgX, GoodsZ)\}$
and the mapping yields the tuple ($\mathcal{M}_{order\_placed(...)}$,$Synch_{order\_placed(...)}$) =

$$(\{m1 = order(AgY, AgX, GoodsZ)\}, \{(m1, t1)\})$$

and the synchronisation indicates that $t1$ is the time at which the $order$ message $m1$ is sent.

For $NormCondition$, the *actors* and *actions* are as follows:

actors($deliver(AgX, AgY, GoodsZ$, 1 week)) = $\{AgX, AgY\}$
actions($deliver(AgX, AgY, GoodsZ$, 1 week)) = $\{deliver(AgX, AgY, GoodsZ)\}$
and the mapping yields the tuple ($\mathcal{M}_{deliver(...)}$,$Synch_{deliver(...)}$) =

$$(\{m2 = notify\_delivery(AgX, AgY, GoodsZ)\}, \{(m2, t1 + 1week\})$$

and the synchronisation indicates that the time at which the $notify\_delivery$ message is sent is within 1 week of the $order$ message $m1$ being sent. ■

Notice that both activation of a norm and the norm conditions may involve multiple actors jointly executing actions according to specific temporal constraints. For example, suppose that the obligation on AgX, to deliver $GoodsZ$ to AgY within 1 week, is activated only if AgY has placed the order, *and* within three days of placing the order AgX receives confirmation from its bank that AgY has cleared monies owed to AgX for previous orders. For this activation condition N' we would have:

($\mathcal{M}_{N'}$,$Synch_{N'}$) =
(
$\{m1 = order(AgY, AgX, GoodsZ)$,
$m2 = notify\_clearance(Bank\_AgX, AgX, debt(AgY))\}$,
$\{(m1, t1), (m2, t1 + 3\text{days})\}$
)

In general, for each normative clause NC in a contract, its $NormActivation$ (NC_A) is mapped to a pair ($\mathcal{M}_{NC_A}$,$Synch_{NC_A}$), which labels a transition to a node (see Figure 2) that denotes a state S which, if the norm is an obligation or prohibition, is critical and so must be monitored. Its $NormCondition$ (NC_C) is mapped to a pair ($\mathcal{M}_{NC_C}$,$Synch_{NC_C}$) that labels a transition from S to S'.

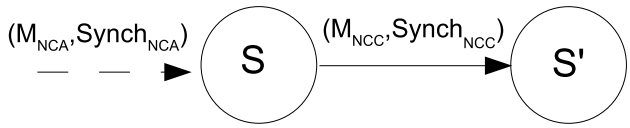

Fig. 2. *ATN* representation of a normative clause

During contract enactment (i.e., at run-time) these ATNs are interpreted with respect to messages observed as defined by the associated synchronisation conditions. In this way, critical states are identified and norm violations detected.

5.2 Interpretation of Augmented Transition Networks

If we refer to Figure 2, then we can make the following general statements:

- If NC is an obligation, then the contract is in a critical state S with respect to NC if messages $\mathcal{M}_{NC_A}$ are observed according to $Synch_{NC_A}$, and the obligation is violated if messages $\mathcal{M}_{NC_C}$ are *not* observed as having been sent according to $Synch_{NC_C}$.

- If NC is a prohibition, then the contract is in a critical state S with respect to NC if messages $\mathcal{M}_{NC_A}$ are observed according to $Synch_{NC_A}$, and the prohibition is violated if messages $\mathcal{M}_{NC_C}$ *are* observed as having been sent according to $Synch_{NC_C}$.

- If NC is a permission, then the contract is in an allowed state S with respect to NC if messages $\mathcal{M}_{NC_A}$ are observed according to $Synch_{NC_A}$, and the permission is executed if messages $\mathcal{M}_{NC_C}$ *are* observed as having been sent according to $Synch_{NC_C}$. We will further motivate requirements for *ATN* representations of permitted behaviours (where the issue of violation does not arise) in Section 5.3.

Example 3. Recall the obligation $\mathbf{Ob}_{Del}$, permission $\mathbf{Per}_{Del}$ and prohibition $\mathbf{Pro}_{Del}$ described in Example 1. Each of these have the same $NormActivation$ and $Norm\ Condition$. Hence for each we obtain the *ATN* shown in Figure 3:

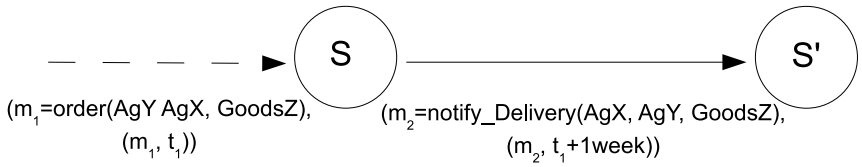

Fig. 3. *ATN* representation of $\mathbf{Ob}_{Del}$, $\mathbf{Per}_{Del}$, and $\mathbf{Pro}_{Del}$

In the case that the *ATN* represents $\mathbf{Ob}_{Del}$ or $\mathbf{Pro}_{Del}$, then the contract is in a critical state with respect to the norm if AgX is observed as having received the message $order(AgY, AgX, GoodsZ)$ from AgY at some time $t1$. If the *ATN* represents $\mathbf{Ob}_{Del}$, then $\mathbf{Ob}_{Del}$ is violated if $notify\_delivery(AgX, AgY, GoodsZ)$ *is not* observed as having been sent by AgX to AgY within 1 week after time $t1$. If the *ATN* represents $\mathbf{Pro}_{Del}$, then $\mathbf{Pro}_{Del}$ is violated if $notify\_delivery(AgX, AgY, GoodsZ)\}$ *is* observed as having been sent by AgX to AgY within 1 week after time $t1$. ■

5.3 Composition of *ATN*s

Thus far we have considered only *ATN*s with two nodes representing individual normative clauses. However, a contract may implicitly specify work-flow patterns by associating the $NormCondition$ of one norm with the $NormActivation$ of another. For

example, AgX's delivery of $GoodsZ$ to AgY, resulting in the contract state S', may itself be an activation condition for another norm. Hence, if the *ATN* in Figure 3 denotes either $\mathbf{Ob}_{Del}$ or $\mathbf{Per}_{Del}$, then S' may denote a state of the contracts in which the obligation:

AgY is obliged to send payment for $GoodsZ$ to AgX within three days

is activated (illustrating why we want to encode *ATN* representations of permitted behaviours). A transition from S' to S'' will then be labelled by a message sent from AgY to AgX indicating payment. If this message is not observed as having been sent in three days, then the obligation will be deemed violated.

Now, suppose the *ATN* in Figure 3 denotes $\mathbf{Pro}_{Del}$. In this case, observation of the sent message $notify\_delivery(AgX, AgY, GoodsZ)$ within 1 week, indicates violation of the prohibition. S' may then denote a critical state of the contract with respect to a now activated secondary obligation — a *contrary to duty* obligation — which now applies to AgX. Such an obligation might be to pay a penalty that is imposed as a punishment by a management party agent that is informed of AgX's violation of $\mathbf{Pro}_{Del}$.

6 Future Work

We have thus far implemented black box agents and their associated communication interfaces as described in Sections 3 and 4. It remains to further specify and implement the mapping mechanisms outlined in Section 5.1 and implement violation detection algorithms (based on matching *ATN*s and observed messages) and violation explanation algorithms for use by the Monitor.

To enable explanation of violations, we may need to refer to some external representation of the the workflow (that is not implicit in the contract). For example, consider an obligation Ob_1 whereby certain behaviours $Per_1, Per2, \ldots$ are permitted in order to realise this obligation (in planning terms the goal state that is obliged to be realised by Ob_1 may be achieved by plans $Per_1, Per2, \ldots$). Certain behaviours $Pro_1, Pro2, \ldots$ may be prohibited from realising Ob_1. If Ob_1 is detected as having been violated, then an explanation may, for example, indicate that Ob_1 was violated because behaviours $Per_1, Per2, \ldots$ were not executed (as determined by the messages corresponding to these behaviours not being observed), and because behaviours $Pro_1, Pro2, \ldots$ were not allowed for realising Ob_1. Notice that such an explanation could be augmented by domain and situation specific information indicating why $Per_1, Per2, \ldots$ could not be executed. For example consider an obligation to repair an aircraft engine within a given time (this example is taken from the CONTRACT prject use case [10]). In order to fulfill this obligation, it may be permitted to source engine parts from one part manufacturer and prohibited to source engine parts from another part manufacturer. If the obligation is violated (no message notifying completion of repair is sent) then the explanation may account for the fact that the permitted ordering of parts did not take place (augmented by situation specific data as to why the permitted behaviour did not occur).

Finally, we note that 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 contract run-time and the intermediate states that are reached prior to violation. These intermediate states can then be explicitly included in the *ATN* representation of contracts, so that during future run-time executions, observation of messages indicating transition to these states may signal preemptive action to avoid violation.

7 Conclusions

We conclude with a discussion of closely related work. As mentioned in the introduction, monitoring of contracts has been extensively studied in a Web Services context (e.g., [6, 12, 13]), where the focus has been on quality of service metrics rather than on the behaviours of contractual entities. Other work has adopted an overhearing approach to monitoring in organisational contexts. Legras et al. [7] use overhearing of messages to monitor changes to the beliefs that agents have about their relationships with other agents in an organisation. Based on this information, a model of how each agent perceives their organisational relationships is accordingly updated. Conversely, Kaminka et al. [5] have developed a plan-recognition approach to overhearing in order to monitor the state of distributed agents that work in a team and collaborate to carry out a specific task. The monitor makes use of the known plan representation of this task to infer on the basis of overheard messages, the belief states of different team-members. These works have thus adopted overhearing in order to infer the mental states of the agents, where these states are domain-dependent and private to the agents. By contrast, in this paper we have described a multi-agent framework that adopts overhearing for a different purpose: it is the states of a contract with respect to its contained norms that are inferred on the basis of messages exchanged. Thus, the proposed approach relies on public knowledge which are norms in a contract. In this way, one can recognise whether a norm is currently in force (activated), in danger of being violated, and whether a norm is in fact violated or complied with. Moreover, in contrast to existing approaches to contract monitoring, our approach benefits from requirements emerging from real world business applications [3].

Acknowledgements The research described in this paper is partly supported by the European Commission Framework 6 funded project CONTRACT (INFSO-IST-034418). The opinions expressed herein are those of the named authors only and should not be taken as necessarily representative of the opinion of the European Commission or CONTRACT project partners.

References

1. R. Conte, R. Falcone, and G. Sartor. Agents and norms: How to fill the gap? *Artificial Intelligence and Law*, 7:1–5, 1999.
2. B. Horling, B. Benyo, and V. Lesser. Using self-diagnosis to adapt organizational structures. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 529–536, Montreal, Canada, June 2001.

3. 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 *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, Industry and Applications Track*, 2008.
4. N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.
5. G.A. Kaminka, D.V. Pynadah, and M. Tambe. Monitoring teams by overhearing: A multi-agent plan-recognition approach. *Journal of Artificial Intelligence Research*, 17:83–135, 2002.
6. A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network Systems Management*, 11(1):57–81, 2003.
7. F. Legras and C. Tessier. Lotto: group formation by overhearing in large teams. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 425–432, New York, NY, USA, 2003. ACM.
8. 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.
9. H. Mazouzi, A. El Fallah Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 517–526, New York,USA, 2002. ACM.
10. 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.
11. S. Miles, N. Oren, M. Luck, S. Modgil, N. Faci, C. Holt, and G. Vickers. Modelling and administration of contract-based systems. In *Proceedings of the AISB 2008 Symposium on Behaviour Regulation in Multi-agent Systems*, pages 19–24. The Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2008.
12. Z. Milosevic, S. Gibson, P.F. Linington, J. Cole, and S. Kulkarni. On design and implementation of a contract monitoring facility. In *Proceedings of the First International Workshop on Electronic Contracting*, page 10. IEEE, 2004.
13. C. Molina-Jimenez, S. Shrivastava, J. Crowcroft, and P. Gevros. On the monitoring of contractual service level agreements. In *WEC '04: Proceedings of the First IEEE International Workshop on Electronic Contracting (WEC'04)*, pages 1–8, Washington, DC, USA, 2004. IEEE Computer Society.
14. C. Molina-Jiménez, S. K. Shrivastava, E. Solaiman, and J. P. Warne. Contract representation for run-time monitoring and enforcement. In *CEC*, pages 103–110, 2003.
15. S. Panagiotidi, J. Vazquez-Salceda, S. Alvarez-Napagao, S. Ortega-Martorell, S. Willmott, R. Confalonieri, and P. Storms. Intelligent contracting agents language. In *Behaviour Regulation in MAS, AISB 2008 Convention Communication, Interaction and Social Intelligence*, pages 49–55, 2008.
16. M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.
17. W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, 1970.
18. L. Xu and M.A. Jeusfeld. Pro-active monitoring of electronic contracts. In J. Eder and M. Missikoff, editors, *Advanced Information Systems Engineering (CAiSE)*, pages 584–600. Springer-Verlag, 2003.