MONITORING COOPERATIVE BUSINESS CONTRACTS IN AN INSTITUTIONAL ENVIRONMENT

Henrique Lopes Cardoso, Eugénio Oliveira
LIACC, DEI / Faculdade de Engenharia, Universidade do Porto, R. Dr. Roberto Frias, 4200-465 Porto, Portugal
hlc@fe.up.pt, eco@fe.up.pt

Keywords: Electronic contract monitoring, Contractual obligations, Deadlines, Rules

Abstract: The automation of B2B processes is currently a hot research topic. In particular, multi-agent systems have been used to address this arena, where agents can represent enterprises in an interaction environment, automating tasks such as contract negotiation and enactment. Contract monitoring tools are becoming more important as the level of automation of business relationships increase. When business is seen as a joint activity that aims at pursuing a common goal, the successful execution of the contract benefits all involved parties, and thus each of them should try to facilitate the compliance of their partners. Taking into account these concerns and inspecting international legislation over trade procedures, in this paper we present an approach to model contractual obligations: obligations are directed from bearers to counterparties and have flexible deadlines. We formalize the semantics of such obligations using temporal logic, and we provide rules that allow for monitoring them. The proposed implementation is based on a rule-based forward chaining production system.

1 INTRODUCTION

Technological support for B2B is increasing. A line of research consists on automating (part of) the process of creation and execution of e-contracts, through multi-agent systems (MAS) technology: software agents can be used as enterprise delegates, automating tasks such as contract negotiation and enactment.

We have formalized and developed an Electronic Institution platform motivated by the need to develop services that assist agents (representing real-world entities) when interacting with the aim of establishing business relationships. Contracts make the commitments of parties explicit, and an institutional environment seeks to monitor and enforce those contracts.

In the B2B world (particularly in cases where a business relationship is strategic) it is often the case that parties cooperate in contract enactment. A contract specifies obligations between contractual parties, and provides legal options handling non-compliance cases. When a business relationship is seen as a joint activity that aims at pursuing a common goal, the successful execution of the contract benefits all involved parties, and thus each of them should try to facilitate the compliance of their partners. In an agent-based environment for contract monitoring, each party is represented in the system by a software agent. Automated monitoring tools should take into account that agents may be cooperative enough to allow, in some circumstances, deviations from their counterparties. This is because group success also benefits each agent's private goals, which are not limited to the ongoing business relationship, but also concern future opportunities that may arise.

Our approach to contract monitoring is based on real-world evidence from business contract legislations, namely the United Nations Convention on Contracts for the International Sale of Goods – CISG (UNCITRAL, 1980), which denotes a flexible and even cooperative facet of trade contracts. In this paper we present the model of contractual obligations that we advocate – *directed obligations with liveline and deadline*. We formalize the semantics of these obligations and provide an implementation concerning the development of an institutional contract monitoring environment.

The paper is structured as follows. In section 2 we introduce our approach to model contractual obli-

gations, and formalize their semantics using temporal logic. In section 3 we translate this formalization to a rule-based approach, and we provide an implementation using a rule engine. An example is given that shows how a model of flexible deadlines can be exploited in a contract. Section 4 concludes.

2 MODEL FOR CONTRACTUAL OBLIGATIONS

When reaching a business agreement, parties sign a contract including norms that describe their commitments. In our approach, an *institutional normative environment* provides a contract monitoring service, including contractual *norms* and monitoring *rules*. Active monitoring requires recording contract-related events in a so-called *normative state*, including:

- institutional facts: institutionally recognized facts that are brought about by agents
- obligations: what agents should do
- fulfillments: obligations that are fulfilled
- violations: obligations that are not fulfilled

Elements other than institutional facts are *environment events*, asserted in the process of norm activation and monitoring. Obligations describe what agents should do, and may be fulfilled or not. We consider different violation states, as explained later.

In general terms, a norm prescribes, given a certain state of affairs, what an agent is obliged to do: *situation* $\rightarrow$ *prescription*. The situation comprises any combination of events that have occurred for a given contract. Consider, for instance, a simple purchase contract. We may have a norm indicating that when the seller issues the invoice (modeled as an institutional fact), the buyer is obliged to pay. Furthermore, we may state that once the payment has been fulfilled, the seller is obliged to send the receipt. We may also define norms based on violations: if the buyer does not pay within due date, an interest rate will be applied.

2.1 Directed Obligations with Time Windows

When specifying norms in contracts, deadline handling is central to define the semantics of contractual obligations[1]. We have developed a model for these obligations inspired in the CISG convention (UNCITRAL, 1980). Consider the following excerpts:

> *Article 48: (1) [...] the seller may, even after the date for delivery, remedy at his own expense any failure to perform his obligations, if he can do so without unreasonable delay [...]; (2) If the seller requests the buyer to make known whether he will accept performance and the buyer does not comply with the request within a reasonable time, the seller may perform within the time indicated in his request. [...]*
>
> *Article 47: (1) The buyer may fix an additional period of time of reasonable length for performance by the seller of his obligations.*
>
> *Article 63: (1) The seller may fix an additional period of time of reasonable length for performance by the buyer of his obligations.*
>
> *Article 52: (1) If the seller delivers the goods before the date fixed, the buyer may take delivery or refuse to take delivery.*

From this source we can see that even though a deadline has been violated, the bearer may still be entitled to fulfill his obligation (Art. 48). Also, the counterparty of an obligation may concede an extended deadline to the bearer (Art. 47 and 63). Finally, anticipated fulfillments are not always welcome (Art. 52).

Based on this background, we define the notion of *directed obligation with liveline and deadline* – $O_{b,c}(f,l,d)$: agent b is obliged towards agent c to bring about f between l (a liveline) and d (a deadline). Directed obligations have been proposed in (Herrestad and Krogh, 1995): obligations are seen as directed from a *bearer* (responsible for fulfilling the obligation) to a *counterparty*. We interpret obligations of this kind as claims from counterparties to bearers (as in (Tan and Thoen, 1998)): if b does not bring about f between l and d then c is *authorized* to react against b. Note that this reaction is discretionary, not mandatory.

The temporal boundaries of obligations define a time window within which the obliged fact should occur. However, not all obligations need livelines. In fact, Art. 52 above is important only in the context of trade transactions, where storage costs may need to be considered. If, on the contrary, a deadline is enough, we can fix l to the moment when the obligation arises. We avoid using a fixed time reference for obligation fulfillment (an obligation for bringing about f at time t), which is suggested in Art. 52; we find it more convenient to define a fixed date as an interval, say, from the start till the end of a specific date.

In order to accommodate the cooperative nature of contract execution, the semantics of directed obliga-

[1] Deontic operators include also permissions and prohibitions, but we find obligations to be particularly relevant in the domain of business contracts.

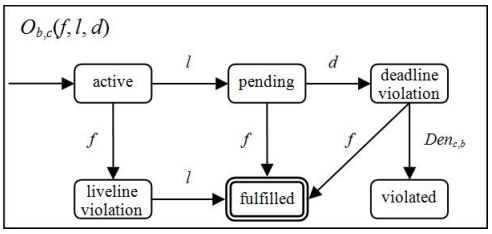

Figure 1: Lifecycle of a directed obligation with liveline and deadline.

tions with liveline and deadline is interdependent with the notion of authorization. We consider that if fact f is not yet the case when deadline d arises, the obligation is not yet violated, but is in a state where the counterparty is authorized to react. We emphasize the case for a *deadline violation* (as opposed to obligation violation). This comprises a flexible approach to handling non-ideal situations: each deadline violation is different, as each may have a different impact on the ongoing business, and occurs between a specific pair of agents with a unique trust relationship.

A deadline is meant to indicate when the counterparty is authorized to react to the non-fulfillment of an obligation directed to him. A possible reaction is to declare the obligation as violated. However, the counterparty might want to concede an extended deadline, by waiting further for the obligation to be accomplished. We introduce the element $Den_{c,b}(f,l,d)$, which is a *denounce* from agent c towards agent b regarding the failure of the latter to comply with his obligation to bring about f before d. We consider the achievement of facts to be common knowledge: a party may only denounce the non-fulfillment of an obligation while that obligation is not fulfilled yet.

Figure 1 illustrates the lifecycle of a directed obligation with liveline and deadline: $O_{b,c}(f,l,d)$. When l arises the obligation becomes pending, since only then it may be fulfilled according to the terms of the contract. In case of an anticipated achievement of f (a liveline violation), we only need l to consider the obligation as fulfilled. However, this does not prevent the counterparty from reacting to this early achievement (although not by denouncing it). A deadline violation can be resolved in two ways: successfully by achieving f, or unsuccessfully by a denounce.

2.2 Formalization

We define the following terms to signal the occurrence of specific obligation states:

- obligation $O_{b,c}(f,l,d)$: the obligation was prescribed by a norm, and is therefore active
- liveline violation $LViol_{b,c}(f,l,d)$: the fact being obliged has been brought ahead of time
- deadline violation $DViol_{b,c}(f,l,d)$: the fact being obliged should have been brought already
- fulfillment $Fulf_{b,c}(f,l,d)$: the obligation was fulfilled
- violation $Viol_{b,c}(f,l,d)$: the obligation was violated and cannot be fulfilled anymore

Borrowing from temporal logic[2], the following relations express the semantics of our obligations:

- $O_{b,c}(f,l,d) \wedge (f\ B\ l) \models LViol_{b,c}(f,l,d)$
  Lifeline violation occurs when the obliged fact is brought about before the liveline.
- $O_{b,c}(f,l,d) \wedge l \wedge (f\ B\ d) \models Fulf_{b,c}(f,l,d)$
  Fulfillment occurs after the liveline and when the obliged fact is brought about before the deadline.
- $O_{b,c}(f,l,d) \wedge (d\ B\ f) \models DViol_{b,c}(f,l,d)$
  Deadline violation occurs when the deadline comes before the obliged fact.
- $DViol_{b,c}(f,l,d) \wedge (f\ B\ Den_{b,c}(f,l,d))$
  $\models Fulf_{b,c}(f,l,d)$
  Fulfillment occurs after a deadline violation if the obliged fact is obtained before a denounce.
- $DViol_{b,c}(f,l,d) \wedge (Den_{c,b}(f,l,d)\ B\ f)$
  $\models Viol_{b,c}(f,l,d)$
  Violation occurs after a deadline violation if a denounce is made before the obliged fact occurs.

We have two kinds of temporal violations: liveline violations – $LViol_{b,c}(f,l,d)$ – and deadline violations – $DViol_{b,c}(f,l,d)$. Note that when the deadline is reached we set the obligation to have a violated deadline, but not to be violated in itself. Only a denounce establishes such a state.

3 AUTOMATED MONITORING RULE-BASED ENGINE

The logical relationships expressed above provide us a formalism to define directed obligations with livelines and deadlines. In order to develop appropriate tools to monitor contracts at run-time, we ground this

[2]We will follow linear temporal logic (LTL) (Emerson, 1990), with a discrete time model. Let $x = (s_0, s_1, s_2, ...)$ be a timeline, defined as a sequence of states s_i. The syntax $x \models p$ reads that p is true in timeline x. We write x^k to denote state s_k of x, and $x^k \models p$ to mean that p is true at state x^k. We use a weak version of the *before* LTL operator B, where q is not mandatory: $x \models (p\ B\ q)$ *iff* $\exists_j(x^j \models p \wedge \forall_{k<j}(x^k \models \neg q))$.

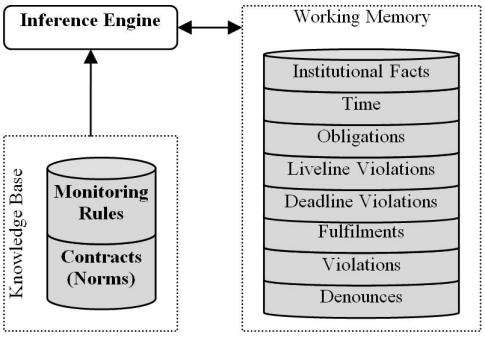

Figure 2: Monitoring as a rule-based production system.

semantics into a reasoning engine capable of responding to events as soon as they occur. A natural choice we have made before (Lopes Cardoso and Oliveira, 2009) is the use of a rule-based forward-chaining production system. These systems are composed of a knowledge base (rules), a working memory (facts) and an inference engine that matches rules' conditions with facts, producing changes in working memory. Figure 2 instantiates these concepts to our purposes.

The following (forward-chaining) rules can be defined to implement the semantics of directed obligations with livelines and deadlines:

- $O_{b,c}(f,l,d) \wedge f \wedge \neg l \rightarrow LViol_{b,c}(f,l,d)$
- $O_{b,c}(f,l,d) \wedge l \wedge f \wedge \neg d \rightarrow Fulf_{b,c}(f,l,d)$
- $O_{b,c}(f,l,d) \wedge d \wedge \neg f \rightarrow DViol_{b,c}(f,l,d)$
- $DViol_{b,c}(f,l,d) \wedge f \wedge \neg Den_{b,c}(f,l,d)$ $\rightarrow Fulf_{b,c}(f,l,d)$
- $DViol_{b,c}(f,l,d) \wedge Den_{c,b}(f,l,d) \wedge \neg f$ $\rightarrow Viol_{b,c}(f,l,d)$

Each relation of the form $(e_1\ B\ e_2)$ was translated into a conjunction $e_1 \wedge \neg e_2$. This allows us to detect the moment at which the *before* relation holds, and consequently to reason about its consequences. We assume an immediate assertion of facts and temporal references when they come into being. Furthermore, rules are expected to be evaluated in every working memory update (e.g. right after a fact is asserted), in order to produce the indicated conclusions. These conclusions are added to the normative state.

3.1 Reasoning with Time

In business contracts deadlines are usually dependent on the fulfillment dates of other obligations. Instead of having fixed (absolute) dates, these may at times be relative, calculated according to other events. CISG (UNCITRAL, 1980) expresses this by saying that dates can be *determinable* from the contract:

> *Article 33: The seller must deliver the goods: (a) if a date is fixed by or determinable from the contract, on that date; (b) if a period of time is fixed by or determinable from the contract, at any time within that period [...]*
>
> *Article 59: The buyer must pay the price on the date fixed by or determinable from the contract [...]*

It is therefore useful to timestamp each event. Therefore, $Fulf_{b,c}(f,l,d)^t$ will indicate a fulfillment at time point t; similarly for $Viol_{b,c}(f,l,d)^t$. Since a fact itself has now a timestamp attribute, for ease of reading we will write fact f achieved at time point t as $Fact(f)^t$. A denounce will be written $Den_{b,c}(f,l,d)^t$.

Norms will be based on these elements and time references in order to prescribe obligations with relative deadlines. For example, $Fulf_{b,c}(Deliver(x),\_,\_)^t \rightarrow O_{c,b}(Pay(price),t,t+10)$ would mean that once agent b has fulfilled his obligation to deliver x to agent c, the latter is obliged to pay to the former within a period of 10 time units.

Having timestamps also allows us to define rules with a closer reading to the LTL *before* operator:

- $O_{b,c}(f,l,d) \wedge Fact(f)^t \wedge t < l \rightarrow LViol_{b,c}(f,l,d)$
- $O_{b,c}(f,l,d) \wedge l \wedge Fact(f)^t \wedge t < d \rightarrow Fulf_{b,c}(f,l,d)^t$
- $O_{b,c}(f,l,d) \wedge d \wedge \neg(Fact(f)^t \wedge t < d)$ $\rightarrow DViol_{b,c}(f,l,d)$
- $DViol_{b,c}(f,l,d) \wedge Fact(f)^t \wedge$ $\neg(Den_{c,b}(f,l,d)^u \wedge u < t) \rightarrow Fulf_{b,c}(f,l,d)^t$
- $DViol_{b,c}(f,l,d) \wedge Den_{c,b}(f,l,d)^u \wedge$ $\neg(Fact(f)^t \wedge t < u) \rightarrow Viol_{b,c}(f,l,d)^u$

This kind of approach has the benefit of relaxing the rule evaluation policy, because we are checking the timestamps of each event (see also (Lopes Cardoso and Oliveira, 2009)).

3.2 Implementation with Jess

We have chosen Jess[3] (Friedman-Hill, 2003) to implement our norm monitoring system. Jess is a very efficient rule engine based on the Rete algorithm for pattern matching. We start by defining appropriate templates (through `deftemplate` constructs) for each type of element in working memory. Jess facts follow a frame-like notation, in which each fact has associated slots to be filled in. We have:

```
(deftemplate ifact
  (multislot fact) (slot when) )

(deftemplate time
  (slot when) )
```

[3]The code presented in this section includes some simplifications in order to make it more simple to understand.

```
(deftemplate obligation
  (slot bearer) (slot counterparty)
  (multislot fact)
  (slot liveline) (slot deadline) )

(deftemplate liveline-violation
  (slot obl) )

(deftemplate deadline-violation
  (slot obl) )

(deftemplate fulfillment
  (slot obl) (slot when) )

(deftemplate violation
  (slot obl) (slot when) )

(deftemplate denounce
  (slot obl) (slot when) )
```

For simplification, we included a reference to the obligation inside other templates. The `time` template is used to assert the occurrence of time events (associated with livelines and deadlines), which is done by scheduling alerts using a system clock.

Implementing the monitoring rules in Jess is straightforward. A Jess rule is written in the form *LHS* => *RHS*, where *LHS* includes fact patterns that will be matched against facts in working memory. The *RHS* indicates actions to execute (such as asserting new facts) when the rule is fired. The following rules (defined with `defrule` constructs) translate directly from the monitoring rules shown above (identifiers starting with a question mark are variables).

```
(defrule detect-liveline-violation
  ?obl <- (obligation (fact $?f) (liveline ?l))
  (ifact (fact $?f) {when < ?l})
  =>
  (assert (liveline-violation (obl ?obl))) )

(defrule detect-fulfillment
  ?obl <- (obligation (fact $?f) (liveline ?l)
             (deadline ?d))
  (time (when ?l))
  (ifact (fact $?f) (when ?t))
  (test (<= ?t ?d))
  =>
  (assert (fulfillment (obl ?obl) (when ?t))) )

(defrule detect-deadline-violation
  ?obl <- (obligation (fact $?f) (deadline ?d))
  (time (when ?d))
  (not (ifact (fact $?f) {when <= ?d}))
  =>
  (assert (deadline-violation (obl ?obl))) )

(defrule detect-belated-fulfillment
  (deadline-violation (obl ?obl))
  ?obl <- (obligation (fact $?f))
  (ifact (fact $?f) (when ?t))
  (not (denounce (obl ?obl) {when <= ?t}))
  =>
  (assert (fulfillment (obl ?obl) (when ?t))) )

(defrule detect-violation
  (deadline-violation (obl ?obl))
  ?obl <- (obligation (fact $?f))
  (denounce (obl ?obl) (when ?u))
  (not (ifact (fact $?f) {when <= ?u}))
  =>
  (assert (violation (obl ?obl) (when ?u))) )
```

These rules enable us to monitor the compliance of agents with prescribed obligations. As explained at the beginning of section 2, norms have a rule-like format and are based on contract-related events, such as those obtained by firing monitoring rules. Not surprisingly, norms too are implemented in Jess as rules. What makes them norms is that typically they will be used to obtain new obligations (these will be asserted in the *RHS* of norms), which will then be monitored.

The normative environment's monitoring capabilities may be used as a tool for alerting agents when certain contract-related events occur. Further rules may be defined with such a purpose. The *RHS* of Jess rules may include function calls that implement the desired level of responsiveness of the normative environment in which notifications are concerned.

3.3 Example

In this section we show a simple example where the concept of flexible deadlines is exploited in an electronically supervised business relationship. We have a contract between two agents, say `B` and `S`, wherein `S` commits to supply, whenever ordered, good `X` for 7.5 per unit. The norms below (implemented as Jess rules) define the contractual relationship and are included in the institutional normative environment for monitoring purposes. Agent `S` is supposed to deliver the ordered goods between 3 to 5 days after the order (norm `n1`), and agent `B` shall pay within 30 days (norm `n2`). Furthermore, if agent `B` does not pay in due time, he will incur in a penalty consisting of an obligation to pay an extra 10% on the order total (norm `n3`). Finally, if agent `S` violates his obligation to deliver, the contract will be canceled (`n4`).

```
(defrule n1
  (ifact (fact order item X quantity ?q)
         (when ?w)))
  =>
  (assert
    (obligation (bearer S) (counterparty B)
                (fact delivery X qt ?q)
                (liveline (+ ?w 3))
                (deadline (+ ?w 5)) ) ) )

(defrule n2
  (fulfillment (obl ?obl) (when ?w))
  ?obl <- (obligation (fact delivery X qt ?q))
  =>
  (assert
```

```
    (obligation (bearer B) (counterparty S)
                (fact payment (* ?q 7.5))
                (liveline ?w)
                (deadline (+ ?w 30)) ) ) )
(defrule n3
  (deadline-violation (obl ?obl))
  ?obl <- (obligation (fact payment ?p)
                      (deadline ?d))
  =>
  (assert
    (obligation (bearer B) (counterparty S)
                (fact payment (* ?p 0.10))
                (liveline ?d)
                (deadline (+ ?d 30)) ) ) )
(defrule n4
  (violation (obl ?obl))
  ?obl <- (obligation (fact delivery X qt ?q))
  =>
  (cancel-contract ID123) )
```

Note that the interest applied on payments is automatic once a deadline violation is detected (norm `n3`). On the other hand, a contract cancellation (norm `n4`) requires that agent `B` denounces the inability of agent `S` to fulfill the delivery. It is therefore up to agent `B` whether to wait further and accept a delayed delivery or not. If the agreed upon contract conditions are important enough, allowing a counterparty deviation (and hence taking a cooperative attitude regarding the compliance of the contract) may be a good decision.

Different kinds of situations may be easily modeled using this kind of norms. Moreover, using flexible deadlines also ensures a degree of freedom for agents to make decisions in the execution phase of contracts, which is important for dealing with business uncertainty.

4 CONCLUSIONS

In B2B relationships contracts specify, through obligations, the interactions between different partners, and provide legal options to which parties can resort in case of conflict. However, when this joint activity aims at pursuing a common goal, the successful performance of business benefits all involved parties. Therefore, when developing automated monitoring tools, one should take into account that agents may be cooperative enough to allow some counterparties' deviations.

In this paper we have introduced a model for contractual obligations. We defined them as directed obligations with livelines and deadlines. The directed aspect concerns the need to identify the agent who will be authorized to react in case of non-fulfillment. Obligation violations are now dependent on the counterparty motivation to claim them.

Our approach is based on real-world evidence from business contracts (namely the United Nations Convention on Contracts for the International Sale of Goods), which denotes a flexible and even cooperative facet of trade contracts. This extends to the concept of B2B Virtual Organizations, wherein different parties come together to share a business goal that is achievable through the cooperative fulfillment of a common contract.

Rule-based production systems allow applying rules in a forward-chaining way. This data-driven approach is appropriate to model a norm monitoring environment based on events. Jess is a powerful rule engine that allows a straightforward implementation of our monitoring model. The easy integration of a Jess engine with a Java application enables the development of a full monitoring tool that aims at supporting a cooperative model of business contracts enactment. We envisage to apply this framework to more complex business scenarios.

ACKNOWLEDGMENTS

The first author is supported by FCT (Fundação para a Ciência e a Tecnologia) under grant SFRH/BD/29773/2006.

REFERENCES

Emerson, E. A. (1990). Temporal and modal logic. In Leeuwen, J. v., editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 995–1072. North-Holland Pub. Co./MIT Press.

Friedman-Hill, E. (2003). *Jess in Action*. Manning Publications Co.

Herrestad, H. and Krogh, C. (1995). Obligations directed from bearers to counterparties. In *Proceedings of the 5th international conference on Artificial intelligence and law*, pages 210–218, College Park, Maryland, United States. ACM.

Lopes Cardoso, H. and Oliveira, E. (2009). A context-based institutional normative environment. In Hubner, J., Matson, E., Boissier, O., and Dignum, V., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems IV*, LNAI 5428, pages 140–155. Springer.

Tan, Y.-H. and Thoen, W. (1998). Modeling directed obligations and permissions in trade contracts. In *Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences, Volume 5*. IEEE Computer Society.

UNCITRAL (1980). United nations convention on contracts for the international sale of goods (cisg).