e-Contract Representation for Defeasible Temporal Reasoning and Conflict Management

Georgios K. Giannikis and Aspassia Daskalopulu

Department of Computer and Communications Engineering,
University of Thessaly,
38221 Volos, Greece
{ggiannik,aspassia}@inf.uth.gr

Abstract. In this report a formal representation and a defeasible reasoning approach for multi-party contractual agreements is presented. Starting from a representation of electronic contracts in Event Calculus, we propose a mapping to a representation in Default Logic. The proposed contract language allows us to reason defeasibly with e-contracts, which is useful in order to determine or explain the normative state of a business exchange in the presence of incomplete or inaccurate knowledge.

Keywords. E-contracts, Non-monotonic reasoning, Event Calculus, Default Logic, Conflict management

September 2006

Table of Contents

1 Introduction

Extensive research, during the last decade, has focused on the establishment and subsequent monitoring of contractual agreements in electronic marketplaces (or so called e-markets) (for example, cf. [13, 1, 12, 19] among many others). Generally, e-contracting is viewed by many researchers as conducted within an e-market, which offers a variety of services, such as: brokering to identify and match prospective business partners; negotiation facilitation; lodging of electronic documents; arbitration and dispute resolution; contract performance monitoring and enforcement. In this report we are concerned with the latter that is with the development of appropriate e-market services for contract performance monitoring and enforcement.

During a business transaction that is regulated by some agreement, the main issues of interest for contract performance monitoring seem to be:

(i) To establish what each party is obliged (or permitted, or prohibited, or empowered and so on) to do at a given point in time;

(ii) To determine whether each party complies with the behaviour stipulated for it by the agreement; where a party deviates from prescribed behaviour—intentionally or due to force majeure—to determine what, if any, remedial mechanisms are applicable that might return the business exchange to a normal course; and

(iii) To detect potential conflicts among parties' obligations (or permissions, prohibitions or powers and so on); when a conflict is detected, to provide a way out through some conflict resolution mechanism.

Many researchers adopt a process view of electronic contracts, that is, they regard them as sets of norms that map out the possible states, in which the actual corresponding business exchanges that are regulated by them may find themselves (for an example, cf. [8]), and some researchers (for example, [18, 3, 9, 15], among others) have adopted Event Calculus [16] to represent and reason with e-contracts. Such representations allow us to address issues (i) and (ii) and partly (iii). However, a representation in Event Calculus does not facilitate reasoning with partial or incomplete knowledge, hypothetical reasoning, and, finally, it does not help towards conflict resolution.

Recently, to address these issues, attention has turned to the deployment of defeasible reasoning (e.g. [12, 11, 22]), using mainly Defeasible Logic [21]. We investigate an alternative approach, in which we use Reiter's Default Logic [25], and propose a mapping from the Event Calculus representation of an e-contract to default rules, extended with priorities [5, 6]. The resulting representation enables us to perform both defeasible deductive reasoning (prediction) and abductive reasoning (explanation). The former allows us to establish what norms are active at a given point in time, based on assumptions and/or knowledge of actions that have already taken place. The latter allows us to explain and justify the norms that are active at a given state of the business exchange. Furthermore, dynamic conflict management, based on a preferential default theory, is also possible. Apart from this, reasoning about the violation of normative propositions and their potential reparations is also facilitated.

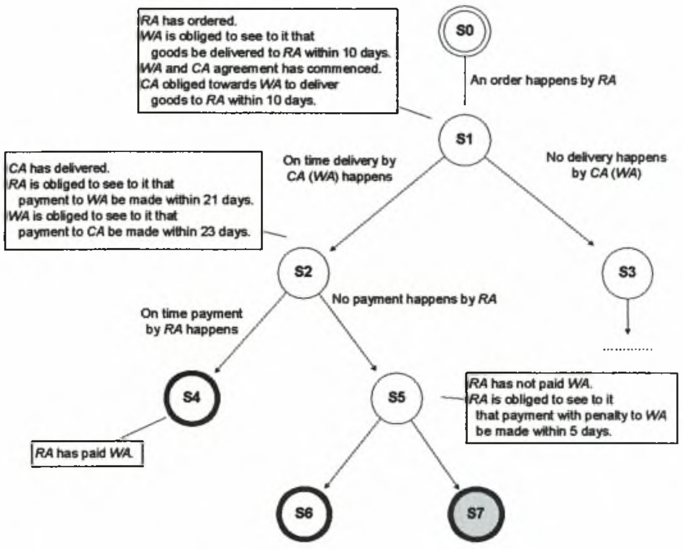
The rest of the report is organized as follows: section 2 introduces an example scenario, which we use for illustration purposes; section 3 discusses briefly the representation of an e-contract in Event Calculus; section 4 shows how the Event Calculus representation may be mapped to default rules and how conflict detection and resolution may be performed; finally, sections 5 and 6 present related work, conclusions and directions for future research.

2 Example Scenario

For the purposes of illustration consider a 3-party business transaction that takes place in an e-market. A retailer requires some goods and for that reason its agent (*RA*) communicates with a wholesaler's agent (*WA*) and establishes an agreement for the provision of such goods. Consequently, the wholesaler's agent (*WA*) communicates with a carrier's agent (*CA*) and establishes an agreement for the timely and safe delivery of the goods from the wholesaler to the retailer. There are two interdependent agreements here, one between *RA* and *WA*, and another between *WA* and *CA*.

The first agreement is to be conducted on the following terms: The *WA* should see to it that the goods be delivered to the *RA* within 10 days from commencement (e.g., the date *RA*'s order takes place). The *RA*, in turn, should see to it that payment be made within 21 days from the date it receives the goods. If the *WA* does not deliver on time, then a fixed amount is to be deducted from the original price of the goods for each day of delay and it should see to it that delivery be made by a new deadline, say within the next 3 days. If the *RA* does not perform payment on time, then a fixed amount is to be added to the original price of the goods for each day of delay and it should see to it that payment be made by a new deadline, say within the next 5 days. In the same spirit, the second agreement between *WA* and *CA* defines obligations, their deadlines and possible reparations in case of violations. Following [8], we may take an informal, process view of the business transaction that is regulated by the agreements. Each state offers a (possibly partial) description of the factual and normative propositions that hold true in it. A transition between states corresponds to an event that takes place, i.e. an action that one of the parties performs or omits to perform. Part of such a description of the business exchange is shown in figure 1.

State *S0* (time point *T0*) is the initial state, no events have occurred yet. Let us assume that the *RA* places an order at some time after *T0*. The transaction is now at state *S1*, where the *WA* is obliged, towards the *RA* to see to it that goods be delivered to the *RA* within 10 days. The *CA*'s obligation towards the *WA* to deliver goods within 10 days is, also, active at state *S1* (time point *T1*). If the *CA* delivers within the specified time bounds, then the business exchange moves to state *S2*, where its obligation (and the dependent *WA*'s obligation towards the *RA*) is successfully discharged, and the *RA*'s obligation towards the *WA* to pay becomes active (as does the *WA*'s obligation to pay the *CA*). If the *CA* does not deliver on time, then the transaction is in state *S3* (time point *T3*), where the *WA* must make amends to the *RA* as specified by their agreement (and the *CA* must make amends to the *WA* as specified by their agreement). Similarly, when the business exchange finds itself at state *S2* (time point *T2*), the *RA*'s obligation towards the *WA* to perform payment within 21 days holds. Compliance with this obligation will lead the exchange to state *S4*, whereas violation will take it to state *S5*. In the same manner we may discuss other states of the business exchange.



S0
An order happens by RA
RA has ordered. WA is obliged to see to it that goods be delivered to RA within 10 days. WA and CA agreement has commenced. CA obliged towards WA to deliver goods to RA within 10 days.
S1
On time delivery by CA (WA) happens
No delivery happens by CA (WA)
CA has delivered. RA is obliged to see to it that payment to WA be made within 21 days. WA is obliged to see to it that payment to CA be made within 23 days.
S2
S3
On time payment by RA happens
No payment happens by RA
S4
RA has paid WA.
S5
RA has not paid WA. RA is obliged to see to it that payment with penalty to WA be made within 5 days.
S6
S7

Fig. 1. Possibly partial contract state diagram

3 A Representation in Event Calculus

The Event Calculus (EC) [16] allows the representation of actions and reasoning about their effects. The basic elements of the language are ***time points***, ***fluents*** and ***actions*** (or else events).

Each state of a business exchange may be associated with a time point and expressions of the form $time_i < time_{i+1}$ indicate the temporal ordering of time points.

Fluents are factual and normative propositions whose truth value alters over time. For our example scenario we use propositions, such as *Order(agent1,agent2)*, *Delivery(agent1,agent2)* and *Payment(agent1,agent2)*, to denote that events/actions of ordering (*AOrder(agent1,agent2)*), delivering (*ADelivery(agent1,agent2)*) and payment (*APayment(agent1,agent2)*) from *agent1* to *agent2* have successfully been performed.

We use expressions of the form *Op*(*agent1,agent2,action,time*) to denote normative propositions that describe that *agent1* is in legal relation *Op* towards *agent2*, to perform *action* by *time*. The legal relation *Op* may be obligation, prohibition or permission. We use these notions as they are understood in Standard Deontic Logic (SDL) [7, 27].

We adopt the simple EC formalism presented in [26], which uses six basic predicates (Table 1) defined in a domain-independent manner.

Table 1. Event Calculus Predicates

Initiates(action,fluent,time)	*fluent* starts to hold after *action* occurs at *time*.
Terminates(action,fluent,time)	*fluent* stops to hold after *action* occurs at *time*.
HoldsAt(fluent,time)	*fluent* holds at *time*.
Happens(action,time)	(instantaneous) *action* occurs at *time*.
Clipped(time1,fluent,time2)	*fluent* is terminated between *time1* and *time2*.
Declipped(time1,fluent,time2)	*fluent* is activated between *time1* and *time2*.

In [20] six domain independent axioms were introduced. We present here only the positive expressions of the *Clipped* and *HoldAt* predicates and later on we purpose an extension for the e-contracting normative domain. In particular, *Clipped* and *HoldsAt* predicates were defined as follows:

$$Clipped(time1,fluent,time2) \equiv \exists action,time[Happens(action,time) \wedge time1 \leq time < time2 \wedge Terminates(agent,fluent,time2)],$$

$$HoldsAt(fluent,time2) \leftarrow [Happens(action,time1) \wedge Initiates(action,fluent,time1) \wedge time1 < time2 \wedge \neg Clipped(time1,fluent,time2)]$$

and

$$HoldsAt(fluent,time2) \leftarrow [\ HoldsAt(fluent,time1) \wedge time1 < time2 \wedge \neg Clipped(time1,fluent,time2)]$$

The agreement between the *RA* and the *WA* may be represented as follows:

$$Initiates(AOrder(RA,WA),Obligation(WA,RA,ADelivery(WA,RA),time+10),time) \leftarrow Happens(AOrder(RA,WA),\ time).$$
$$Initiates(ADelivery(WA,RA),Obligation(RA,WA,APayment(RA,WA),time+21),time) \leftarrow Happens(ADelivery(WA,RA),time).$$
$$Termnates(ADelivery(WA,RA),Obligation(WA,RA,ADelivery(WA,RA),time+10),time) \leftarrow Happens(ADelivery(WA,RA),\ time).$$
$$Termnates(APayment(RA,WA),Obligation(RA,WA,APayment(RA,WA),time+21),time) \leftarrow Happens(APayment(RA,WA),\ time).$$

Marin *et al.*, [18] distinguish between the so called internal time of norms (e.g. the deadline for an obligation to be met), which in our representation is the time that appears in the normative propositions of the form *Op(agent1,agent2,action,time)* and the so called external time of norms, i.e. the time at which a normative proposition comes into force or ceases to hold.

In our example, initially, the following is true:

$\neg$ *HoldsAt(Order(RA,WA), T0).*

Then, suppose that the *RA* places an order by time *T<T1*. Using the representation above and the definition of the *HoldsAt* predicate we may infer that:

HoldsAt(Order(RA,WA),T1)
HoldsAt(Obligation(WA,RA,ADelivery(WA,RA),T+10),T1)

which corresponds to state *S1*. In similar spirit, suppose that *WA* delivers goods by time *T'<T2*, so

HoldsAt(Delivery(WA,RA),T2)
HoldsAt(Obligation(RA,WA,APayment(RA,WA),T'+21),T2)

which corresponds to state *S2*. Finally, suppose that *RA* pays *WA* on time, that is

HoldsAt(Payment(RA,WA),T4)

is true in state *S4*.

3.1 The effects of Actions in Norm-governed Settings

In [17] it is noted that the effects of an action apply only when the action is considered valid. Whether an action is considered valid depends on whether its actor has:
(i) the legal ability (power, [14]) to perform it,
(ii) permission to exercise this power, and
(iii) practical ability to perform the action.

We consider legal ability as introduced in [14] that is institutionalized power to perform an action. The term permission refers to the normative notion of permission as defined by SDL. As argued in [17, 14], although the notion of institutionalized power is close to the deontic notion of permission, it is imperative to consider them as two distinct notions. This argument can be easily substantiated with a simple example in relation with our case study scenario. Assume a retailer's agent that is legally empowered with the ability to perform payment on behalf of the contractual party it represents. But, the same agent is not permitted to perform payment if the transferred amount of money overcomes a specific value (i.e. 10.000 euros). Based on this example, we consider that the agent has the institutionalized power to perform an action as a general rule. Exceptions to this rule are also applicable by defining permissions that probably apply only at specific time points, time periods or occasions. We should note that an exception is not practically able to block an agent to perform an action, i.e., the action's outcome will count as valid. As Artikis argued in [3] an action is considered as valid, or in other words effective, if the agent had the institutionalized power to perform it. Later on, we enhance this definition by concluding one more prerequisite for valid actions.

We use expressions of the form *IPower(agent,action)* to denote that *agent* is institutionally empowered to perform *action*. We also use expressions of the form *PAblity(agent,action)* to denote that *agent* is physically/practically capable of performing *action*.

We have slightly modified the original definition of the *Happens* predicate to take these points into account and formulate rules that characterize actions as performable (or not), valid (or not), and legal (or not). In our representation the *Happens* predicate includes the actor of the action that occurs, i.e. it takes the form *Happens(agent,action,time)*.

Action is performable/possible by its performer at a given *time*, if *agent* has the practical ability to perform it at that *time*:

$$\begin{aligned} &Possible(agent,action,time) \equiv HoldsAt(Possible(agent,action),time) \equiv \\ &\qquad \exists agent,action,time[HoldsAt(PAbility(agent,action),time)] \end{aligned}$$

For example, if an agent is practically capable of accepting the delivery of some products we may conclude that the action of delivery is possible to happen. But what is the case, in the e-marketplace, if the receiver was not authorized for that action. Shall we count the action of delivery as possible to occur or as occurred?

Therefore, *action* is only considered valid, in the sense of being effective/countable in the overall framework, if *agent* is legally empowered and physically able to perform it:

$$\begin{aligned} &Valid(agent,action,time) \equiv HoldsAt(Valid(agent,action),time) \equiv \\ &\qquad \exists agent,action,time[HoldsAt(IPower(agent,action),time) \\ &\qquad \wedge HoldsAt(Pability(agent,action),time) \end{aligned}$$

Back to our example, if the agent that receives products is, capable and authorized, then we may conclude that the obligation of delivery has been successfully met.

Additionally, *action* is considered legal, if it is valid and no specific forbiddance for its performer is explicitly stated:

$$\begin{aligned} &Legal(agent,action,time) \equiv HoldsAt(Legal(agent,action),time) \equiv \\ &\qquad \exists agent,action,time[HoldsAt(Valid(agent,action),time) \\ &\qquad \wedge \neg\ (\ HoldsAt(\neg Permission(agent,action),time)) \vee \\ &\qquad HoldsAt(Forbiddance(agent,action),time)\ )] \end{aligned}$$

For example, the doorman of a company is empowered to receive all packages except those that are explicitly sent to the company director. This is the case where an exception, in the form of a forbiddance, is in force. If the doorman receives a package while he is not permitted to do so, then this will count as a valid action, but the doorman will have violated the company's internal rule.

Consequently, we have modified Shanahan's definitional axioms (for *Clipped* and *Declipped* predicates) and principles (for *HoldsAt* and ¬*HoldsAt* predicates) by including the performing party as an input parameter and the notions of institutionalized power, permission and practical ability. For instance, the *Clipped* and *HoldsAt* predicates are now defined as follows (the other two axioms change accordingly):

$$\begin{aligned}Clipped(time1,fluent,time2) \equiv \exists agent,action,time[&Happens(agent,action,time)\\ &\wedge time1 \leq time < time2 \wedge Terminates(agent,fluent,time2)\\ &\wedge HoldsAt(Valid(agent,action),time)]\end{aligned}$$

and

$$\begin{aligned}HoldsAt(fluent,time2) \leftarrow [&Happens(agent,action,time1) \wedge Initiates(action,fluent,time1)\\ &\wedge time1 < time2 \wedge \neg Clipped(time1,fluent,time2)\\ &\wedge HoldsAt(Valid(agent,action),time)]\end{aligned}$$

Now, the *HoldsAt* predicate means that *fluent* is true at *time2* if *action* occurred by a legally empowered and practically able *agent* at an earlier time point *time1*, this action initiated *fluent* and the fluent has not been set to false during the interval between *time1* and *time2*.

During this work, we consider that initially all partiers are practically empowered to perform actions in order to fulfil their obligations. We follow this approach because practical inability mainly concerns exogenous parameters, such as the absence of the product receiver or the lack of network/bank account or others in a similar sense that cannot be predicted. In the same way, we treat permissions. According to the common sense law of inertia, an agent is not permitted or is forbidden to perform an action, only if there is an explicit predicate that supports it. Regardless of whether this is the case or not, continuous checking of what is permitted or not cannot block the effects of actions that have occurred, but it can only point out illegal actions.

On the other hand, the continuous verification procedure of what is valid and respectively what is legal is imperative. This is due to the fact that invalid actions cannot affect the state of a transaction. As a result, we are only concerned with actions that can affect the current state of the electronic institution, and those actions are the ones that are valid irrespectively of whether they are legal or illegal.

3.2 Contrary to Duty Structures

Another point worth mentioning concerns the so called Contrary-To-Duty structures (CTDs) [24]. CTDs arise when a primary obligation is defined for a party, along with a rule that determines a secondary obligation for it, should the primary one be violated. For instance, in our example, the *WA* is obliged to deliver within 10 days from the date the *RA*'s order is placed. If it does not do so, then it is obliged to deliver within the next 3 days and to claim a reduced price.

$$\begin{aligned}&HoldsAt(Obligation(agent1,agent2,raction,time3),time2)\\&\wedge\neg HoldsAt(Obligation(agent1,agent2,action,time1),time2)\\&\qquad\leftarrow[HoldsAt(Obligation(agent1,agent2,action,time1),time1)\\&\qquad\wedge\neg Happens(agent1,action,time)\\&\qquad\wedge time\leq time1<time2<time3\\&\qquad\wedge HoldsAt(Valid(agent1,action),time)]\end{aligned}$$

The CTD axiom states that, if *agent1* had an obligation to perform *action* up to a specific time point and, furthermore, this action was valid in the sense that the agent had the legal and practical ability to perform it, in order to meet its obligation, but it was not performed till the deadline, then this obligation ceases to hold and a new obligation, holds.

We should note, that during this work it is not our purpose to analyse all possible cases of CTD structures as presented in [24]. We do not address matters relating to the persistence of norms or indeed periodicity. We assume that when primary obligations are violated, some reparation action (*raction*) may be specified, for instance reparation delivery (*RADelivery*) and reparation payment (*RAPayment*).

Another point worth mentioning is that the notions of legal and practical power to perform an action have a key role in CTD structures. For instance, during the case of a contract violation from an agent that was not empowered to perform the obligatory action the contract representation formalism should provide an alternative reparation taking into account the fact that the agent was not able to meet its obligations. Similar to the previous case, we consider all possible cases under the term of reparation action (*raction'*), so the derived axiom follows the form shown below:

$$\begin{aligned}&HoldsAt(Obligation(agent1,agent2,raction',time3),time2)\\&\wedge\neg HoldsAt(Obligation(agent1,agent2,action,time1),time2)\\&\qquad\leftarrow[HoldsAt(Obligation(agent1,agent2,action,time1),time1)\\&\qquad\wedge\neg Happens(agent1,action,time)\\&\qquad\wedge time\leq time1<time2<time3\\&\qquad\wedge\neg HoldsAt(Valid(agent1,action),time)]\end{aligned}$$

As can be observed, we may reformulate the two CTD axioms in one that is shorter and more general than the initial ones. We do not use such a shorthand, because by using two separate axioms we may distinguish explicitly between reparations that arise due to deviations from prescribed behaviour as a result of the lack of practical or institutional ability, and reparations that arise as a result of violations that occur due to other reasons.

3.4 Contract Representation Language

Based on what were mentioned so far, we are able to derive an EC-based contract representation first order language L_A (FOL) that consists of well-formed formulae (wff) over an alphabet A. The alphabet A consists of variables denoted by lowercase letters such as *action,fluent,time,agent,...*, primitive predicates denoted with initial capital letters as the ones shown in the following list:

- *Initiates(action,fluent,time)*, denoting that *fluent* starts to hold after *action* occurs at *timet*.
- *Terminates(action,fluent,time)*, denoting that *fluent* stops to hold after *action* occurs at *time*.
- *HoldsAt(fluent,time)*, denoting that *fluent* holds at *time*.
- *Happens(agent,action,time)*, denoting that instantaneous action *action* is being occurred by *agent* at *time*.
- *Clipped(time1,fluent,time2)*, denoting that *fluent* is terminated between time points *time1* and *time2*.
- *Declipped(time1,fluent,time2)*, denoting that *fluent* is activated between time points *time1* and *time2*.
- *AOrder(agent1,agent2)*, denoting that *agent1* orders form *agent2*.
- *ADelivery(agent1,agent2)*, denoting that *agent1* delivers to *agent2*.
- *APayment(agent1,agent2)*, denoting that *agent1* pays *agent2*.
- *Order(agent1,agent2)*, denoting that *agent1* ordered form *agent2*.
- *Delivery(agent1,agent2)*, denoting that *agent1* delivered to *agent2*.
- *Payment(agent1,agent2)*, denoting that *agent1* paid *agent2*.
- *Obligation(agent1,agent2,action,time)*, denoting that *agent1* is obliged against *agent2* to perform *action* until *time*.
- *Permission(agent,action)*, denoting that *agent* is permitted to perform *action.*
- *Forbiddance(agent,action)*, denoting that *agent* is prohibited to perform *action.*
- *PAbility(agent,action)*, denoting that *agent* is practically empowered/has the physical power to perform *action*.
- *IPower(agent,action)*, denoting that *agent* is institutionalized empowered/has the institutionalized power to perform *action*.
- *Possible(agent,action)*, denoting that *action* is performable by *agent.*
- *Valid(agent,action)*, denoting that *action* is concerned as valid when performed by *agent.*
- *Legal(agent,action)*, denoting that *action* is concerned as legal when performed by *agent.*

and constants such as *RA, WA,...* for agents and *T,T1,T2,...* for time points. Moreover, it consists of logical constants such as $\neg$ for classical negation, $\wedge$ for conjunction, $\vee$ for disjunction and $\leftarrow$ for implication.

A variation of the simple EC, enhanced with new predicates that specify new domain dependent definitions/axioms and new domain independent axioms, is finally adjusted as follows:

$$\begin{aligned} &Clipped(time1,fluent,time2) \equiv \exists action,time[Happens(agent,action,time) \\ &\qquad \wedge time1 \leq time < time2 \wedge Terminates(agent,fluent,time2) \\ &\qquad \wedge HoldsAt(Valid(agent,action),time)] \end{aligned} \tag{1}$$

$$\begin{aligned} &Declipped(time1,fluent,time2) \equiv \exists action,time[Happens(action,time) \\ &\qquad \wedge time1 \leq time < time2 \wedge Initiates(agent,fluent,time2) \end{aligned} \tag{2}$$

$$\wedge HoldsAt(Valid(agent,action),time)]$$

$$\begin{aligned}&Possible(agent,action,time)\equiv\\&HoldsAt(Possible(agent,action),time)\equiv\\&\qquad\exists agent,action,time[HoldsAt(PAbility(agent,action),time)]\end{aligned}\tag{3}$$

$$\begin{aligned}&Valid(agent,action,time)\equiv\\&HoldsAt(Valid(agent,action),time)\equiv\\&\qquad\exists agent,action,time[HoldsAt(IPower(agent,action),time)\\&\qquad\wedge HoldsAt(Pability(agent,action),time)\end{aligned}\tag{4}$$

$$\begin{aligned}&Legal(agent,action,time)\equiv\\&HoldsAt(Legal(agent,action),time)\equiv\\&\qquad\exists agent,action,time[HoldsAt(Valid(agent,action),time)\\&\qquad\wedge\neg(\ HoldsAt(\neg Permission(agent,action),time))\vee\\&\qquad HoldsAt(Forbiddance(agent,action),time)\ )]\end{aligned}\tag{5}$$

$$time1<time2\tag{6}$$

$$\begin{aligned}&HoldsAt(fluent,time2)\\&\qquad\leftarrow[Happens(agent,action,time1)\wedge Initiates(action,fluent,time1)\\&\qquad\wedge time1<time2\wedge\neg Clipped(time1,fluent,time2)\\&\qquad\wedge HoldsAt(Valid(agent,action),time1)]\end{aligned}\tag{7}$$

$$\begin{aligned}&\neg HoldsAt(fluent,time2)\\&\qquad\leftarrow[Happens(agent,action,time1)\wedge Terminates(action,fluent,time1)\\&\qquad\wedge time1<time2\wedge\neg Declipped(time1,fluent,time2)\\&\qquad\wedge HoldsAt(Valid(agent,action),time1)]\end{aligned}\tag{8}$$

$$\begin{aligned}&HoldsAt(Obligation(agent1,agent2,raction,time3),time2)\\&\wedge\neg HoldsAt(Obligation(agent1,agent2,action,time1),time2)\\&\qquad\leftarrow[HoldsAt(Obligation(agent1,agent2,action,time1),time1)\\&\qquad\wedge\neg Happens(agent1,action,time)\\&\qquad\wedge time\leq time1<time2<time3\\&\qquad\wedge HoldsAt(Valid(agent1,action),time)]\end{aligned}\tag{9}$$

$$\begin{aligned}&HoldsAt(fluent,time2)\leftarrow[\ HoldsAt(fluent,time1)\\&\qquad\wedge time1<time2\wedge\neg Clipped(time1,fluent,time2)]\end{aligned}\tag{10}$$

$$\begin{aligned}&\neg HoldsAt(fluent,time2)\leftarrow[\ \neg HoldsAt(fluent,time1)\\&\qquad\wedge time1<time2\wedge\neg Declipped(time1,fluent,time2)]\end{aligned}\tag{11}$$

Axioms (1) - (5) are definitional ones, which determine the effects of actions on fluents or on the overall institutionalized framework. Axioms (7) and (8) correspond to direct effects of occurring events and thus the shifting from one state to the other. Axiom (9) is a contrary to duty structure that reassigns obligations to contractual parties. Axioms (10) and (11) express the common sense law of inertia, expressing the persistence of fluents during the absence of events that influence their values.

3.5 Comments

A representation in Event Calculus, allows us to establish what each party is obliged (or permitted, forbidden, empowered) to do at a given time point. It also allows us to determine whether each party complies with the agreement, and what, if any, reparatory mechanisms are stipulated, should violations arise. This may be achieved through appropriate queries on the *HoldsAt* predicate. We may, also, spot potential conflicts, for example if such a query returns that a particular agent is both obliged and forbidden to perform a specific action at the same time.

What we cannot do is reason based on incomplete and partial knowledge, or based on assumptions and, moreover to retract previous conclusions in the presence of new knowledge. For example, in the absence of explicit knowledge about legal or practical ability, permission or prohibition to perform actions or even changes in the world, parties may assume that the optimal (for them) conditions hold, during the contract performance phase, and plan their actions accordingly. But later in the presence of new information, parties should be able to update their beliefs and retract previous conclusions. For instance, suppose we are the *RA* agent. In the absence of knowledge to the contrary, at a given time point, we may assume that *CA* is able to deliver goods on time, and based on this, we may infer the exact time point our obligation to pay will be in force. Suppose that later on, we find out that *CA* cannot deliver because nobody (on our part) was present to take the delivery. This new fact should change our view of the world, and any inferences we made so far. Also, although we may spot potential conflicts, by examining what normative propositions hold at a given state, we have no way to resolve them dynamically. We now turn our attention to these issues.

4 Defeasible Reasoning with e-Contracts

Defeasible reasoning allows for non-monotonic, inference with incomplete/uncertain knowledge based on assumptions and, when enhanced with priorities, conflict management. In this section, we describe a mapping from the EC representation to default rules, where we adopt Reiter's Default Logic (DfL) [25] and Brewka's priority settlement between default rules [5, 6].

A default rule has the form:

$$P{:}J_1,J_2,\ldots J_n/C$$

where P is the *prerequisite*, $J=\{J_1, J_2, \dots J_n\}$ is a set of *justifications* and C is the derived *consequent*. The semantics of this inference rule is: If P holds and the assumption J is consistent with our current knowledge, then C may be inferred. Defaults of the form $P{:}C/C$ are called *normal*. A Default Theory (DfT) is a pair of the form (D,W), where W is a set of closed wffs that hold, and are called as background knowledge, while D is a set of defaults. A rule is applicable to a set of formulae $E \subseteq W$ if and only if $P \in E$ and $\neg J_1, \dots, \neg J_n \notin E$ [25]. In this case, the set E is called *extension* of the default theory. Extensions are the most complicated concept of Reiter's default theory because it is hard to determine an accurate belief set for which justifications should be consistent. In Reiter's initial paper for DfL [25] three important properties of extensions were referred. In particular, an extension E of a default theory (D,W):

(i) should contain W,

(ii) should be deductively closed and

(iii) for a default rule of the form $P{:}\ J_1, J_2, \dots J_n / C$, if $P \in E$ and $\neg J_1, \dots, \neg J_n \notin E$ then $C \in E$.

In this work we consider a grounded DfT, that is a theory where defaults contain no free variables and we derive extensions in the manner presented in [2].

4.1 Translation Schema

We may map our EC representation onto default rules. The specific mapping depends on what information is available in the knowledge base about a particular domain. For example, the definition of the *HoldsAt/2* predicate may correspond to the following default:

$$\frac{Happens(agent,action,time1) : \begin{array}{l} Initiates(action,fluent,time1), \\ time1 < time2, \\ \neg Clipped(time1,fluent,time2), \\ HoldsAt(Valid(agent,action),time1) \end{array}}{HoldsAt(fluent,time2)}$$

It may also be seen as the following default rule:

$$\frac{\begin{array}{l} Happens(agent,action,time1) \\ \wedge\, Initiates(action,fluent,time1) \end{array} : \begin{array}{l} time1 < time2, \\ \neg Clipped(time1,fluent,time2), \\ HoldsAt(Valid(agent,action),time1), \end{array}}{HoldsAt(fluent,time2)}$$

or even as :

$$\frac{\begin{array}{l}Happens(agent,action,time1)\\ \wedge Initiates(action,fluent,time1)\\ \wedge\neg Clipped(time1,fluent,time2)\end{array} : \begin{array}{c}time1<time2,\\ HoldsAt(Valid(agent,action),time1)\end{array}}{HoldsAt(fluent,time2)}$$

A fourth possibility is to view it as a normal default, that is:

$$\frac{\begin{array}{l}Happens(agent,action,time1)\wedge time1<time2\\ \wedge Initiates(action,fluent,time1)\\ \wedge\neg Clipped(time1,fluent,time2)\\ \wedge HoldsAt(Valid(agent,action),time1)\end{array} : HoldsAt(fluent,time2)}{HoldsAt(fluent,time2)}$$

Knowledge that we want to be proved from the knowledge base is mapped to the prerequisites part of the default rule, while knowledge that is absent from the knowledge base and may be assumed is mapped to the justifications part of the default rule. The mapping to normal defaults is, of course, stricter. We are currently investigating ways in which the mapping from the EC representation to defaults may be (semi)automated.

Before mentioning the reasoning approaches and their applications, we should define a DfT for the contract domain in respect with the L_A language presented in section 3. Thus, a default contract theory is a pair of *(D,W)* where *W* is a set of FOL facts and strict if-then rules and *D* is a set of default rules. We consider three classes of default rules in respect with their use. The first one is default rules as presented in Reiter's initial work and above. The second class are again strict rules that do not belong in the belief set *W*, but in the *D* set. Default strict rules can be represented as justification-free defaults of the form *P: /C*. Finally the third class is the so called priority rules. Priority rule comes to enhance the default theory with priorities that stand among classical defaults. Brewka in [5, 6] first introduced Prioritised Default Logic and we follow some of those early ideas in order to provide dynamic prioritized conflict management to our contractual domain as shown below.

To sum up, the proposed default contract theory contains:

- *Facts* as presented in the EC formalism, For example: *Holds(Order(RA,WA),T)*
- *Strict rules*, also in respect with the EC language and represented either as predicate logic implication rules or as DfL justification-free defaults. It is worth noting that retractable reasoning is not possible in the first case.
- *Default rules* as presented above.
- *Priority rules* whose conclusion is a priority relation that stands among defaults as noted below. Priority rules can be formulated either as strict rules or as defaults.

We should also note down that exceptions, as they were described in section 3, can be easily and expressively represented as default rules with justifications. For example, we can enhance axiom (7) with a justification to represent the agent's permission (or forbiddance) to perform an action.

$$\frac{Happens(agent,action,time1) : \begin{array}{c} Initiates(action,fluent,time1), \\ time1 < time2, \\ \neg Clipped(time1,fluent,time2), \\ HoldsAt(Valid(agent,action),time1), \\ HoldsAt(Permission(agent,action),time1) \end{array}}{HoldsAt(fluent,time2)}$$

4.2 Deductive Reasoning

DfL enables us to reason with incomplete knowledge, by deriving conclusions that are based on consistent assumptions, which may be retracted later, in the presence of new information. There are two approaches to perform such inference. In the first one, the *sceptical reasoning*, a formula is entailed by a default theory, if it is derived by all its extensions. This is a strict approach and requires the computation of all possible extensions and subsequent check to determine if a formula belongs in all of them. We may adopt this approach to implement a planning and advisory tool that could be used during the contract formation phase. In the second approach, the *credulous reasoning*, a formula is entailed by a default theory, if it is derived by at least one extension. Such an approach might be useful to implement a 'what-to-do-next' tool, which could be used during the contract enforcement phase, to support decision-making when violations or conflicting obligations arise. We adopt the operational definition of extensions of [2], as explained below, which uses Reiter's original definition of extensions and derives them by maintaining sets of formulae.

Given a closed DfT *(W,D)* and considering a finite or infinite set of defaults *DS=(DR1, DR2,...)* from *D* extensions are easily derived by formulating two sets. The first one, denoted as *In(DS)*, is populated by what was initially believed and everything that is concluded when adding a new default in the DS set. We should note that a default can only be applied once. The second set, denoted as *Out(DS)*, is populated with formulae that were assumed to be false (the negation of justifications formulae) and that finally should not turn out to be true. According to this approach the set *In(DS)* is an extension of the default theory iff *DS* is *successful* and *closed*. The two properties of the DS set are defined as follows: DS is successful iff *In(DS)* and *Out(DS)* have no formula in common *(In(DS)∩Out(DS)=0)*, while *DS* is closed iff every applicable default of the default theory already occurs in *DS*.

For example, consider the default theory *(W,D)* with:

$W=\{\neg HoldsAt(Order(RA,WA), T0), Happens(RA,AOrder(RA,WA),T), T0<T\}$

and *D* includes the following closed defaults (*DR1*, *DR2*, *DR3* respectively):

$$\frac{\begin{array}{c}\mathit{Happens(RA,AOrder(RA,WA),T)}\\ : \\ \mathit{T0 < T < T1 < T3},\\ \mathit{Initiates(AOrder(RA,WA),Obligation(WA,RA,ADelivery(WA,RA),T3),T)},\\ \neg \mathit{Clipped(T,Obligation(WA,RA,ADelivery(WA,RA),T3)},\\ \mathit{HoldsAt(Valid(RA,AOrder(RA,WA)),T)}\end{array}}{\mathit{HoldsAt(Obligation(WA,RA,ADelivery(WA,RA),T1)}}$$

$$\frac{\begin{array}{c}\mathit{Happens(RA,AOrder(RA,WA),T)}\\ : \\ \mathit{Initates(AOrder(RA,WA),Causes(AOrder(RA,WA),AOrder(WA,CA)),T)},\\ \mathit{HoldsAt(Valid(RA,AOrder(RA,WA)),T)}\end{array}}{\mathit{HoldsAt(Causes(AOrder(RA,WA),AOrder(WA,CA)),T)}}$$

$$\frac{\begin{array}{c}\mathit{Happens(RA,AOrder(RA,WA),T)}\\ \wedge\, \mathit{HoldsAt(Causes(AOrder(RA,WA),AOrder(WA,CA)),T)}\\ : \\ \mathit{HoldsAt(Valid(RA,AOrder(RA,WA)),T)},\\ \mathit{HoldsAt(Valid(WA,AOrder(WA,CA)),T)}\end{array}}{\mathit{Happens(WA,AOrder(WA,CA),T)}}$$

For the purposes of illustration we use a special predicate called *Causes/2*. The formulae *Causes(action1,action2)* denotes that the occurrence of *action1 causes action2* to occur or in other words that *action2* is being *caused* by *action1*.

DS={DR1}, *DS={DR2}*, *DS={DR1, DR2}* or *DS={DR2, DR3}* are successful but not closed because *(DR2, DR3)*, *(DR3, DR1)*, *(DR3)* and *(DR1)*, respectively, are also applicable. *DS={DR1, DR2, DR3}* is closed and successful and therefore it may be considered as an extension. Moreover, consider that the above default theory is enhanced with the normal default *DR4*:

$$\frac{\mathit{true} : \neg \mathit{HoldsAt(Valid(WA,AOrder(WA,CA)),T)}}{\neg \mathit{HoldsAt(Valid(WA,AOrder(WA,CA)),T)}}$$

In this case if *DR4* applies before *DR2* or *DR3* then its consequent ¬*HoldsAt(Valid(WA,AOrder(WA,CA)),T)* will block the firing of *DR3*. Although a causal relation holds and consequently an indirect event should fire this specific exception blocks its occurrence.

4.3 Abductive Reasoning

DfL can also be used for abductive reasoning and serve as the basis for a tool that explains the factual and normative propositions that hold at various contractual states. The problem takes the form: given a formula *F* and a default theory *(W,D)*, can we prove that *F* can be explained? Reiter proposed an algorithm for this kind of (backward) reasoning, based on linear resolution [25], which is also discussed in detail in Poole [23].

Consequently, we only point out the outlines of abduction using closed normal defaults. What we really search for is a set $DS \subseteq D$ such that from $W \cup cons(D)$ we may imply *F*, where *cons(DS)* is the set of consequents of all rules that are present in *DS*. The algorithm, in short, for proving that formula *F* is valid is:

- put $\neg F$ in a form where only disjunction relations exists and derive a finite set of literals representing those disjunctions (for example, from the $\neg A \wedge B$ formula by applying De Morgan's law comes up $A \vee \neg B$ that is equal to the set $\{A, \neg B\}$)
- take a set of clauses *C* in conjunction normal form (CNF) which results form the $W \cup cons(D)$ set and apply linear resolution
- repeat the same procedure trying to explain the preconditions of each default that were used in the proof
- if eventually we get $\perp$ (refutation) and $W \cup cons(D)$ is consistent that means that $\neg$F is not satisfied, or in other words that F is valid; otherwise F is not valid

Consider, now the example shown in the previous section on deductive reasoning and reformulate all defaults to turn out as normal defaults. Can we explain *Happens(WA,AOrder(WA,CA),T)* from the default theory *(W,D)*, where *W={¬HoldsAt(Order(RA,WA), T0), Happens(RA,AOrder(RA,WA),T),* $T_0<T$*, ... all other formulae related with, validness and value's initiation/persistence...)* and *D={DR1,DR2,DR3}* in their normal form? In this case $\neg F \equiv$*Happens(WA,AOrder(WA,CA),T)* and *C={¬HoldsAt(Order(RA,WA), T0), Happens(RA,AOrder(RA,WA),T),* $T_0<T$*, ... all other formulae related with, validness and value's initiation/persistence..., {HoldsAt(Obligation(WA,RA,ADelivery(WA,RA), T3),T1),DR1}, {Holds(Causes(AOrder(RA,WA),AOrder(WA,CA)),T),DR2}, {Happens(WA,AOrder(WA,CA),T),DR3}}*. By applying linear resolution in order to prove *F* we have that $W \cup cons(DR1,DR2,DR3)$ and $W \cup cons(DR2,DR3)$ are two possible sets that are consistent and proofs for *Happens(WA,AOrder(WA,CA),T)*.

4.4 Conflict Detection

We are interested in developing a mechanism that may facilitate agents to deal with conflicts that may arise during a business transaction, i.e. to detect them and, if possible, to resolve them.

An agent faces a potential conflict if the transaction is in a state, where contradictory normative propositions hold. For instance consider a state, where two propositions of the following form hold:

$$HoldsAt(Obligation(agent1, agent2, action, time2), time1)$$

HoldsAt(¬Obligation(agent1, agent2,action,time3),time1)

During the interval ΔT=[*time1*,min*(time2,time3)*], the agent cannot decide whether to satisfy the norm stipulated by the first proposition and do the *action* that is obligated or to behave in accordance with the second norm, which essentially permits it either to do *action* or not to do *action* (and do something else) or to remain idle.

Another case where an agent faces a potential conflict is where the transaction is in a state where two propositions of the following form hold:

HoldsAt(Obligation(agent1, agent2,action,time2),time1)

HoldsAt(Obligation(agent1, agent2,¬action,time3),time1)

During the interval ΔT, the agent faces the dilemma whether to do *action* or to do *¬action*; we do not make, at present, any attempt to characterize formally what is meant by "negative action", and treat informally such an expression as meaning "not doing *action* but doing something else instead, or doing nothing".

A third possibility for an agent to face a conflict is where two propositions of the following are true in a state:

HoldsAt(Obligation(agent1, agent2,action,time2),time1)

HoldsAt(Obligation(agent1, agent3,action',time3),time1)

Here *agent1* bears two obligations (possibly for two distinct actions) towards two distinct agents. Assuming that no parallel action is allowed, during the interval ΔT the agent cannot decide which to fulfill.

There are, of course, other ways in which conflicts may arise, but we do not discuss this issue here any further. What we want to focus on, instead, is how such conflicts may be detected.

A DfL representation of contracts, such as the one we propose, allows us to detect conflicts by examining extensions, which are essentially set of propositions. In general, a potential conflict arises when there are multiple extensions of a default theory that represents a contract, and one of them contains a proposition that conflicts with a proposition contained in another; let us call these *inter-extension* conflicts. Conflicts may also arise even when there is a single extension of the default theory, if it contains conflicting propositions; let us call these *intra-extension* conflicts.

The detection of inter-extension conflicts is useful for an agent, which finds itself in a state that is not, *yet*, problematic, and has alternative courses of action to consider. The agent must decide upon a specific course of action – some way of *preventing* the potential conflicts from ever arising is required. The detection of intra-extension conflicts, on the other hand, essentially informs the agent that it is, *already*, in a problematic state. Again the agent needs a way to *resolve* the conflict and decide which norm to satisfy in a way that minimizes the damage done – since, unavoidably, some norm will be violated.

In the next section we propose a unified way for conflict prevention and conflict resolution.

4.5 Conflict Management: Prevention and Resolution

In [5, 6], Brewka proposes a Prioritised Default Theory (PDfT) that enables us to define and apply priorities on default rules dynamically. A PDfT is a triple *(W,D,name)*, where *name* is a function that assigns names to default rules *D*. The extension of a PDfT is derived in the same way as in a DfT.

What makes PDfTs really useful is that we may reconsider the priorities of the default rules, when new conclusions are derived as a result of a rule firing. Using dynamic priorities, we generate preferred extensions, where each extension indicates a transaction plan. Priorities amongst ground defaults may be defined dynamically by specifying domain-dependent conditions of interest.

The general pattern for ascribing priorities dynamically takes the form of a default rule:

$$prule(d'',[d,d'],v) \equiv \frac{\begin{array}{l} rule(d,v) \wedge rule(d',v) \\ \wedge HoldsAt(condition(v),time) \end{array} : HoldsAt(d < d',time)}{HoldsAt(d < d',time)}$$

Here *d, d', d''* are variables that denote names of ground defaults; *prule(d'',[d,d'],v)* is a label denoting a priority-assignment default rule *d''*, which is applied on the defaults *d, d'* and prioritizes them based on some condition that is checked for some set of entities of interest *v*; *rule(d,v)* denotes a ground default *d* and its set of entities of interest *v*. The intended interpretation of this rule is: if two defaults *d* and *d'* apply and some criterion is satisfied at *time* then *d* takes priority over *d'* from that time onwards, if this may be consistently assumed.

In this manner, we may manage conflicts in a variety of ways, by specifying different criteria (e.g. obligations should be satisfied in the order in which they arise; or obligations towards specific agents should be satisfied first etc). We should note that, we treat priorities between defaults as fluents, because priority assignment is driven by criteria and these may themselves be event-driven. Where some criterion is event-independent (e.g. ordering of time points) it need not be treated as a fluent. The following examples illustrate these points.

In the first example, assume that we have two retailers *RA* and *RA'* that order goods from the same wholesaler *WA*, but at different time points ($T<T'<T1$). Consequently, from the *WA*'s perspective, at state *S1* (time point *T1*) two obligations for delivery hold. Suppose that only one action may be performed at any given time. In this case, the *CA* agent (who is commissioned by the *WA*) has two conflicting obligations to satisfy and two default extensions are derived (*E* and *E'* respectively). An inter-extension conflict arises:

$$rule(DR1,T) \equiv$$

$$\frac{\begin{array}{l} Happens(RA, AOrder(RA,WA),T) \wedge T < T1 < T3 \\ \wedge\, Initiates(AOrder(RA,WA), \\ \qquad\qquad Obligation(CA,WA,ADelivery(CA,RA),T3),T) \\ \wedge\, \neg Clipped(T,Obligation(CA,WA,ADelivery(CA,RA),T3),T1) \\ \wedge\, HoldsAt(Valid(RA,AOrder(RA,WA)),T) \end{array} : HoldsAt(Obligation(CA,WA,ADelivery(CA,RA),T3),T1)}{HoldsAt(Obligation(CA,WA,ADelivery(CA,RA),T3),T1)}$$

and

$$rule(DR2,T') \equiv$$

$$\frac{\begin{array}{l} Happens(RA', AOrder(RA',WA),T') \wedge T' < T1 < T3 \\ \wedge\, Initiates(AOrder(RA',WA), \\ \qquad\qquad Obligation(CA,WA,ADelivery(CA,RA'),T3),T') \\ \wedge\, \neg Clipped(T,Obligation(CA,WA,ADelivery(CA,RA'),T3),T1) \\ \wedge\, HoldsAt(Valid(RA',AOrder(RA',WA)),T') \end{array} : HoldsAt(Obligation(CA,WA,ADelivery(CA,RA'),T3),T1)}{HoldsAt(Obligation(CA,WA,ADelivery(CA,RA'),T3),T1)}$$

Now, suppose that we use the following normal default rule (*PR1*) to assign time-based priorities:

$$\frac{\begin{array}{l} rule(dr,time) \wedge rule(dr',time') \\ \wedge\, time \leq time' \end{array} : HoldsAt(dr < dr',time')}{HoldsAt(dr < dr',time')}$$

This rule gives priority to *DR1* over *DR2* and consequently *E* is the preferred extension. The effect is that obligations are met in the order in which they arise.

In the second example, assume that the carrier agent did not deliver goods to any of the two ordering agents for some reason. In this case two rules apply. The two potential extensions contain conflicting obligations. According to (*PR1*), *E* is again the preferred extension. But, what would happen, if we had in our initial knowledge base a strict rule (*PR2*) such as the following:

$$\begin{array}{l} HoldsAt(dr' < dr,time') \leftarrow HoldsAt(PC(agent'),time') \\ \qquad\qquad\qquad\qquad\qquad \wedge\, rule(dr,agent) \wedge rule(dr',agent') \end{array}$$

This rule defines a priority between defaults based on specific information, in this case, that some specific agent is a privileged customer (*PC(agent')*). In this case, the previous default rule (*PR1*) that led to extension *E* is not applicable, because inconsistency with the consequent of the new rule arises. So, according to (*PR2*) the preferred extension is *E'*, and the business transaction will follow an alternative course than the one chosen before.

5 Related Work

There are many research attempts that were based on Event Calculus as a temporal formalism to represent and reason about actions and states of contractual agreements and normative systems. In [18] an analysis of certain temporal aspects of law was presented. This work was mainly concerned with the distinction of the so called internal and external time of legal norms. Also an extension of the EC was proposed in order to overcome some of its limitations. Artikis et al in [3] introduce a representation in EC of open computational agent societies. This work is concerned with notions of computational systems, such as social constraints, roles and states, and of normative social systems, such as institutionalized power, obligation and permission. Also a society visualiser was presented that executes a variation of the Contract Net Protocol (CNP). Farrell et al in [9] proposed an EC representation of contracts to support automatic state tracking of normative states. Obligation, permission and institutionalized power are the main normative concepts this works concerns with. In order to describe the effects, on normative states, of the events that occur during the transaction, an ecXML representation based on the XML version of the EC was also introduced. In [15] a simple EC representation of contracts was presented. The representation was considered in the Belief-Desire-Intention (BDI) architecture and deals with two types of contract's, short and long contracts, monitoring and execution. Contracts violations are being overcome by associating meta-information with them. This fact enabled agents to solve potential violations and met their goals by outsourcing. Furthermore an agent architecture in the style of AgentSpeak(L) was also described. Those approaches do not, however, address defeasible reasoning.

There are three approaches to defeasible reasoning with e-contracts, besides ours: Grosof's [12], Governatori's [11] and Paschke's [22]. Grosof's SweetDeal is based on the SweetRules prototype. This work represents contract's rules via Situated Courteous Logic Programs (SCLP) that is an extension of Ordinary Logic Programs (OLP) with prioritized conflict handling and procedures to perform actions and queries on contractual states. Mutual exclusion statements that were added in the knowledge base support conflict detection while priorities statements appoint the resolving mechanism. Governatori's and Hoang's DR-Contract architecture extends DR-Device architecture [4] by using defeasible deontic logic of violations (DDLV). DDLV was presented in [10] and combines deontic notions with defeasibility and violations. Based on Nute's Defeasible Logic theory [21], four kinds of knowledge are adopted: facts; strict rules; defeasible rules, i.e., rules that can be defeated by other rules; superiority relation that defines priority among rules. A special kind of operator was also introduced in order to represent CTD structures. ContractLog, introduced by Paschke et al, addresses a special kind of contracts called Service Level Agreements (SLAs). Those are agreements that mainly contain provisions for maintaining service quality. This approach combines various theoretical approaches and finally proposes a formal representation for contractual agreements together with monitoring and enforcement tools. Specifically, an SLA is represented by Event-Condition-Action (ECA) rules that were enhanced with predicates from EC, for reasoning about actions and their effects, and other predicates for deontic notions. A special kind of ECA rule was also introduced in order to represent CTD structures. Moreover, three types of conflicts were defined: authorization, obligation and application specific conflicts. Conflict resolution was addressed i) with the adoption of the four basic types of knowledge as were proposed in Nute's Defeasible Logic theory and by defining static priorities among rules and furthermore ii) by considering ***conditional mutexes*** as were proposed in Grosof's Generalized Courteous Logic Programs (GCLP). ContractLog were implemented based on Mandarax rule engine and the Prova language extension and supports both deductive and abductive reasoning. All three approaches propose a mapping of their normative rules formalism on RuleML (XML) in order to integrate their frameworks with Semantic Web Technology.

Of the three approaches, only ContractLog allows for temporal reasoning. Moreover, in all three, the priorities used to resolve conflicts are statically defined. Finally it is not clear, in all three, whether and how abductive inference may be supported.

6 Conclusions and Future Work

In this report an EC contract representation and a defeasible reasoning approach for multi-party contractual agreements were presented. The proposed EC formalism gives us the ability to reason about actions and their effects and about normative propositions that may get activated on potential states of the transaction as the outcome of the occurring events. DfL, the logic formulation for defeasible reasoning, gives us the ability to make retractable decisions based on assumptions and also helps us to explain normative relations that are observed on different contractual states. EC and DfL were related by the proposed mapping schema from the initial form of rules into default rules. Defaults were also enhanced with priorities that modulate a dynamic decision taking mechanism. Moreover, our approach considers conflicts between contractual normative propositions and proposes a conflict management framework that is based on priority settlement and the above mentioned dynamic and defeasible decision making mechanism.

We are currently investigating (i) how conflicts may be more precisely characterized; and (ii) how the translation from EC to DfL may be (semi)automated. We also want to examine in detail how our approach, which uses Reiter's Default Logic, compares with those that are based on Nute's Defeasible Logic. Finally, we plan the development of a tool, which may be integrated with Semantic Web technologies.

Acknowledgments

This work was supported by the European Social Fund (75%) and the Greek Secretariat for Research and Technology (25%) through the 3rd Community Support Programme - Measure 8.3 (PENED2003-03EΔ466).

References

[1] Tom Allen and Robin Widdison. Can computers make contracts? *Harvard Journal of Law and Technology*, 9(1), 1996.

[2] Grigoris Antoniou. A tutorial on default logics. *ACM Computer Surveys*, 31(4):337–359, 1999.

[3] Alexander Artikis, Jeremy Pitt, and Marek J. Sergot. Animated specifications of computational societies. In *AAMAS*, pages 1053–1061. ACM, 2002.

[4] Nick Bassiliades, Grigoris Antoniou, and Ioannis P. Vlahavas. DR-Device: A defeasible logic system for the semantic web. In Hans Jörgen Ohlbach and Sebastian Schaffert, editors, *PPSWR*, volume 3208 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2004.

[5] Gerhard Brewka. Reasoning about priorities in default logic. In *AAAI*, pages 940–945, 1994.

[6] Gerhard Brewka and Thomas Eiter. Prioritizing default logic. In *Intellectics and Computational Logic, Papers in Honor of Wolfgang Bibel*, Applied Logic Series 19, pages 27–45. Kluwer Academic, 2000.

[7] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[8] Aspassia Daskalopulu. Modeling legal contracts as processes. In *Legal Information Systems Applications, 11th International Workshop on Database and Expert Systems Applications (DEXA'00)*, pages 1074–1079. IEEE Computer Society, 2000.

[9] Andrew D. H. Farrell, Marek J. Sergot, Mathias Sallé, and Claudio Bartolini. Using the event calculus for tracking the normative state of contracts. *Int. J. Cooperative Inf. Syst.*, 14(2-3):99–129, 2005.

[10] Guido Governatori. Representing business contracts in RuleML. *Int. J. Cooperative Inf. Syst.*, 14(2-3):181–216, 2005.

[11] Guido Governatori and Duy Hoang. A semantic web based architecture for e-contracts in defeasible logic. In *RuleML*, pages 145–159, 2005.

[12] Benjamin N. Grosof. Representing e-commerce rules via situated courteous logic programs in RuleML. *Electronic Commerce Research and Applications*, 3(1):2–20, 2004.

[13] John Ibbotson and Marty Sachs. Electronic trading partner agreement for e-commerce. Technical report, IBM Corporation, 1999.

[14] Andrew J.I. Jones and Marek J. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):427–443, 1996.

[15] John Knottenbelt and Keith Clark. Contract-related agents. In Francesca Toni and Paolo Torroni, editors, *CLIMA VI*, volume 3900 of *Lecture Notes in Computer Science*, pages 226–242. Springer, 2005.

[16] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95, 1986.

[17] David Makinson. On the formal representation of rights relations. *Journal of Philosophical Logic*, 15(4):403–425, 1986.

[18] Rafaél Hernández Marín and Giovanni Sartor. Time and norms: a formalisation in the event-calculus. In *ICAIL*, pages 90–99, 1999.

[19] Michael Merz, Frank Griffel, M. Tuan Tu, Stefan Möller-Wilken, Harald Weinreich, Marko Boger, and Winfried Lamersdorf. Supporting electronic commerce transactions with contracting services. *Int. J. Cooperative Inf. Syst.*, 7(4):249–274, 1998.

[20] Rob Miller and Murray Shanahan. The event calculus in classical logic - alternative axiomatisations. *Electron. Trans. Artif. Intell.*, 3(A):77–105, 1999.

[21] Donald Nute. Defeasible logic. In Dov Gabbay, Christopher J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3, pages 353–395. Oxford University Press, 1994.

[22] Adrian Paschke, Martin Bichler, and Jens Dietrich. ContractLog: An approach to rule based monitoring and execution of service level agreements. In *RuleML*, pages 209–217, 2005.

[23] David Poole. Default logic. In Dov Gabbay, Christopher J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3, pages 189–215. Oxford University Press, Oxford, 1994.

[24] Henry Prakken and Marek J. Sergot. Contrary-to-duty obligations. *Studia Logica*, 57(1):91–115, 1996.

[25] Raymond Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1-2):81–132, 1980.

[26] Murray Shanahan. The event calculus explained. In *Artificial Intelligence Today*, pages 409–430. 1999.

[27] Georg H. von Wright. Deontic logic. *Mind, Oxford University Press*, 60:1–15, 1951.