8th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 25 – 27, 2006

EXCEPTIONS IN CONTRACT IN DEFEASIBLE LOGIC

Ioan Alfred LETIA, Adrian GROZA
Technical University of Cluj-Napoca
Department of ComputerScience
Baritiu 28, RO-400391 Cluj-Napoca,
{letia,adrian}@cs-gw.utcluj.ro

Abstract. *Although contracts are a central mechanism for defining interactions between firms, there is currently inadequate business support for using the information provided by these contracts. Our work deals with contract's exceptions or violations, with the goal to automate the dispute resolution between business partners. Both, the representation of contracts and agents reasoning modules are captured by defeasible logic using Deimos. The framework includes also OAA agents responsible for: computing penalties, providing default rules and immutable rules, supporting the dispute resolution through argumentation.*
Keywords: *electronic contracts, dispute resolutions, defeasible reasoning, agents, contract law.*

1. Introduction

Contracts are a central mechanism for defining interactions between firms, but there is currently inadequate business support for using the information provided by these contracts, including both static contract definitions or real-time execution. The current requirements of the supply chain demand a more outward view on contract management for each entity within the chain. Having such an accurate perspective on the contract data, the parties implied could manage optimally the exceptions that may appear during contract validity.

Our work handles such contract's exceptions or violations, with the goal to automate the dispute resolution between business partners. The design of punishment policies applied to specific domains linking agents' actions to material penalties is an open research issue. The emergence of online dispute resolution (ODR) mechanisms is due to the new challenges imposed by the global character of e-commerce. The technology applied to ODR should contribute to the settle of the dispute. We use defeasible logic framework to provide a place for computation in which we describe how conflicting norms are processed. Our work is a step forward through the automation of the dispute resolution for a B2B context. Reasoning about intentions and other mental attitudes has a defeasible nature, and defeasibility is one of the proper characteristics of normative inference. The agents are built in the Open Agent Architecture[1] and they employ defeasible logic [4,7,6] in order to model the contractual clauses. The reasoning mechanism with these clauses is done with Deimos[2].

This paper is structured as follows. In the next section we present defeasible logic and in section 3 we introduce contracts within the task dependency network model. Section 4 describes the architecture of the system and the defeasible agents within it. Section 5 browses some related work and finally, section 6 concludes the paper.

2. Defeasible logic

The defeasible theory we use (a knowledge base in defeasible logic, or a defeasible logic program) [4, 7, 6] consists of five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation.

Facts are indisputable statements, for example, "The advertisePrice for item is 1000", or "The promotion lasts 30 days". These two facts would be expressed as:

$\rightarrow$*advertisePrice(item, 1000).*
$\rightarrow$ *promotion(item, 30days).*

Strict rules are rules in the classical sense: whenever the premises are indisputable (e.g. facts), then so is the conclusion. An example of a

[1] http://www.ai.sri.com/~oaa/
[2] http://www.cit.gu.edu.au/~arock/defeasible/Defeasible.cgi

strict rule is “A client with 10 orders is considered a regular business partner":

$nrOrders(P, 10) \rightarrow regularPartner(P).$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is “Regular business partners usually have discounts":

$regularPartner(P) \Rightarrow discount(P).$

The idea is that if we know that a customer is a regular one then we may accord him discount, unless there is other, not inferior, rule suggesting the contrary.

Defeaters are rules that cannot be used to draw any conclusions. Their only use is to prevent some conclusions. An example is: “If the customer is a regular one and he has a short delay for paying, we might don't ask for penalties". This rule cannot be used to support a “not penalty" conclusion, but it can prevent the derivation of the penalty conclusion:

$regularPartner(P), delay(P, short) \rightsquigarrow \neg penalty(P).$

The *superiority relation* among rules is used to define priorities, that is, where one rule may override the conclusion of another rule. For example, given the defeasible rules:

$r : nrOrder(P, 100) \Rightarrow regularPartner(P).$

$r': \quad lastOrder(P, \quad 1month)) \quad \Rightarrow \neg regularPartner(P).$

which contradict one another, no conclusive decision can be made about whether a client is a regular one. If we introduce a superiority relation $>$, and $r' > r$ with the intended meaning that r' is strictly stronger than r, we can indeed conclude that a client that didn't order anything within last month, should not be considered a regular one. All rules r have a consequent $C(r)$ and a finite set of antecedents $A(r)$. We denote by R_s the set of strict rules in a theory, by R_d the set of defeasible rules, and their union by R_{sd}. A conclusion of a defeasible theory D is a tagged literal that may be proved by D, and can have one of the following forms:

$+\Delta q$, meaning that q is definitely provable in D (fig.1);

$+\Delta$:
If $P(i+1) = +\Delta q$ then
$\exists r \in R_s[q]\ \forall a \in A(r) : +\Delta a \in P(1..i)$

$-\Delta$:
If $P(i+1) = -\Delta q$ then
$\forall r \in R_s[q]\ \exists a \in A(r) : -\Delta a \in P(1..i)$

Figure 1. Definitely (not definitely) provable.

$-\Delta q$, meaning that q is not definitely provable in D (fig.1);

$+\partial q$, meaning q is defeasibly provable in D (fig.2);

$-\partial q$, meaning that q is not defeasibly provable in D (fig.3).

A derivation $P= P(1),...,P(n)$ is a finite sequence of tagged literals, satisfying the conditions in figures 1, 2, and 3. Figure 1 states the conditions for concluding whether a query is definitely provable. At step $i+1$ one can assert that q is definitely provable if there is a strict rule r with consequent q and all the antecedents of r have been asserted to be definitely provable in previous steps. Similarly, the conclusion that q is not definitely provable can be reached at step $i + 1$ if all strict rules having as consequent q have at least one antecedent previously asserted as not definitely provable.

A goal q, which is not definitely provable, is defeasibly provable if we can find a strict or defeasible rule r having q as a consequent for which all antecedents are defeasibly provable, ~q is not definitely provable, and for every rule having q as a consequent we can find an antecedent which does not satisfy the defeasibly provable condition.

$+\partial$:
If $P(i+1) = +\partial q$ then either
(1) $+\Delta q \in P(1..i)$ or
(2) (2.1) $\exists r \in R_{sd}[q] \forall a \in A(r)$
$+\partial a \in P(1..i)$ and
(2.2) $-\Delta \sim q \in P(1..i)$ and
(2.3) $\forall s \in R[\sim q]$
$\exists a \in A(s) : -\partial a \in P(1..i)$

Figure 2. Defeasibly provable.

$-\partial$:

If $P(i+1) = -\partial q$ then

(1) $-\Delta q \in P(1..i)$ and

(2) (2.1) $\forall r \in R_{sd}[q]\ \exists a \in A(r)$ $-\partial a \in P(1..i)$ or

(2.2) $+\Delta \sim q \in P(1..i)$ or

(2.3) $\exists s \in R[\sim q]$ such that $\forall a \in A(s) : +\partial a \in P(1..i)$

Figure 3. Not defeasibly provable.

A goal *q* is not defeasibly provable if it is not definitely provable, and one of the following conditions holds: all strict or defeasible rules implying *q* have a body term which is not defeasibly provable, or ~*q* is definitely provable, or there is a rule having ~*q* as a consequent and all its antecedents are defeasibly provable. An important property of defeasible logic is that it is coherent, that is, there is no defeasible theory D and literal *p* such that *D* |- +∂*p* and *D* |- -∂*p* or *D* |- +Δ*p* and *D* |- -Δ*p* (we cannot establish that a literal is simultaneously provable and unprovable).

3. Problem Specification

3.1 Contracts Representation

Contracts specify obligations for the signatories, as well as their permissions and prohibitions, and may state penalties in cases where these policies are violated. They may also state rewards for outstanding performance [9]. The first step in processing a contract written in natural language is to provide its logical representation. All the clauses of the contract are transformed into facts, definitions and normative rules. A contract is a compound object. It contains an informative section *I* and a behavioural specification *B*, which are common knowledge for each involved party. The informative section *I*=<*id, s, b, def,* t_{issue}, $t_{maturity}$, *R*> consists of: a contract number that uniquely identifies the contract for the parties involved (*id*); the mappings between identities and roles (the seller *s*, the buyer *b*); definitions of some contract terms (*def*); the contract validity period (start date or t_{issue}, expiration date or $t_{maturity}$); the normative system of reference (*R*). Here, *R* can be the imposed remedy: expectation damage, reliance damage, opportunity costs [2]. The behavioural specification is a set of normative statements describing the expected behaviour of the roles defined in the informative section: *B*=<*pricePolicy, order, delivery, payment, termination, disputes*>. Because it would be impossible and costly for the parties to specify in the behavioural specification all the consequences of every possible exception, the law provides default rules, which fill in the gaps of the actual contract. The normative agent that appears in rule r_1 from fig. 4 provides such default rules. In this case, when an exception arises and no remedy is explicitly specified, the victim asks the *expectation-damages-agent* for a default rule. We provide such normative agents (see section 4.2) that implement different principles of contract law. These agents encapsulate a smaller, but important class, of "immutable rules", which are rules that parties cannot change by any contractual agreement and they govern even if the parties have overwrote them. If one party did not agree with the solution computed by the normative agents, it can ask the dispute resolution agents (see section 4.3) for a new solution.

3.2 Task dependency network

We model the contracts signed between agents in order to form the supply chain by adapting the task dependency network model [11] used in the analysis of the supply chain's exceptions as follows: task dependency network is a directed, acyclic graph, (V, E), with vertices $V = G \cup A$, where: G = the set of goods, $A = S \cup P \cup C$ the set of agents, S = the set of suppliers, P = the set of producers, C = the set of consumers, and a set of edges E connecting agents with their input and output goods. With each agent $a \in A$ we associate an input set I_a and an output set O_a: I_a= {$g \in G$ | $<g,a> \in E$} and O_a= {$g \in G$ | $<a,g> \in E$}. The agent *a* is a supplier if $I_a = 0$, a consumer if $O_a = 0$, and a producer in all other cases. Without any generalization lost, we consider that a consumer $c \in C$ needs a single item ($|I_c| = 1$) and every supplier $s \in S$ or producer $p \in P$ build one single item ($|O_s| = 1$ and $|O_p| = 1$). An agent must have a contract for all of its input goods in order to

produce its output, named *presumable* [3] and denoted by $\hat{p}$. If we note $n_p = |I_p|$, the agent has to sign n_p+1 [4] contracts in order to be a member in the supply chain. For each input good $g_k \in I_p$ the agent p bids its own item valuation v_k. The auction for the good g_k sets the transaction price at p_k. The agent's investments are $I_p = \sum_{k=1}^{n_p} p_k$, where k are the winning input goods. We note by I_p^g the agent's investments, without considering the investments made for the current good g. Similarly, we note all bids values submitted by the agent p as $V_p = \sum_{k=1}^{n_p} v_k^p$ and this value without considering the bid for good g as V_p^g. For the output good, the agent p signs a contract at reliance price R_p.In the task dependency network model, the contractual agent extracts the set O of statements where he is the subject and decides whether to fulfill them or not. violated.

//Informative specification
id: *0045.*
s: *myrole(seller).*
b: *partner_role(buyer).*
def: *currency(EU), jurisdiction(EU).*
t_{issue} $t_i(27.05.2006)$.
$t_{maturity}$ $t_i(29.05.2006)$.
R: r_1: *breach* $\Rightarrow$ *expectation-damages-agent*
//Behavioural specification
price: r_2: *regularPartner(P)* $\Rightarrow$ *discount(P)*
r_3: *specialOrder(X)* $\Rightarrow \neg$*discount(X)*
$r_2 > r_3$.
order: r_4: *item(http://item.xml)*
r_5: *currency(X)* $\Rightarrow$ *price(http://priceX.xml)*
r_6: *shipTime(X)* $\Rightarrow$*notification(X+5).*
delivery r_7: *ship(3days)*
r_8: *delay(X)* $\Rightarrow$ *penalty(10+X)*
payment r_9: *shipTime(X)* $\Rightarrow$ *pay(X+2)*
r_{10}: *delay(X)* $\Rightarrow$ *interest(1%)*
termination r_{11}: *replacement(0045,newContract)*
r_{12}: *frustration.*
disputes r_{13}: *quality* $\Rightarrow$ *ODR-argumentation-agent*
r_{14}: *payment* $\Rightarrow$ *ODR-mediation-agent*

Figure 4. Contract sample.

[3] Note that when someone breaches a contract with a presumable agent it has to pay more damages.
[4] Suppliers and consumers have to sign one contract only.

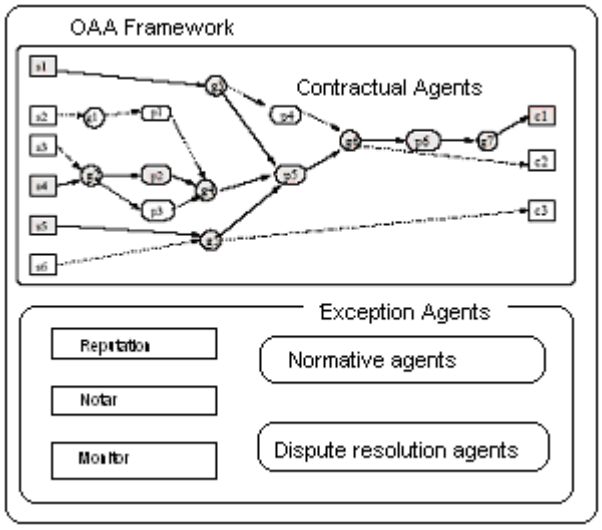

Figure 5. System architecture.

The agent extracts R, the set of all the rights (normative statements where the agent is beneficiary) granted to him. For every normative statement in R, the agent decides either to ask the counterpart to start the fulfillment of the norm or to signal the counterpart or the normative agents that the norm has been breached.

4. Argumentation System

Our agents are developed in the Open Agent Architecture and they use Deimos for the defeasible reasoning. The framework contains contractual agents, which represent business partners and some "exception agents" which deal with the breaches of the contracts (see fig. 5): notary, monitor, reputation, definitions, templates, normative agents (expectation damages, reliance damages, opportunity costs), and dispute resolution agents (argumentation, mediator, arbitrator).

4.1 Contractual agents

The contractual agents need to be able to determine at any point which actions are permitted, how soon the obligatory actions are required, and how severe the penalties could be. They base their decisions on their beliefs, which are influenced by the behavior of the other agents and by the data coming from various enterprise systems. In our approach, obligations are no longer absolute, but are relative to their

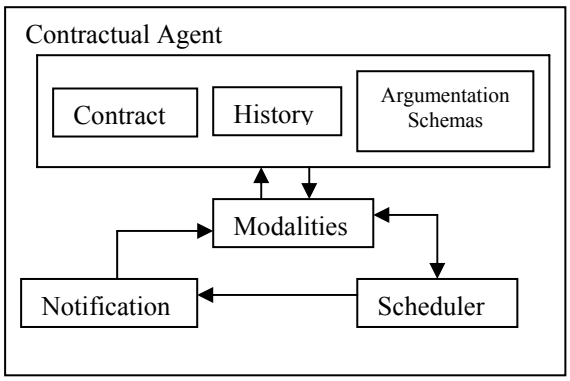

Figure 6. Contractual agent.

associated sanctions. This extension gives space for deliberative decisions by the agent as whether to breach or not a contractual clause. The notification module is responsible for the communication with other agents. The modalities module updates the agent's beliefs and selects the normative statements to fulfill. The scheduler manages the execution of the various normative statements based on their priorities. The history module contains information about past interactions with the current partner. For instance, it stores how many orders it had already done. Using the above information and the clauses from the contract, the agent infers that the partner is a regular one, for example, and it builds the context attached to the contract.

4.1.1 Reasoning

From the contract specification, the agents derive their obligations or permissions. For instance, if the seller has shipped the item, but he does not receive any payment yet, he usually considers this as a violation: r_1: *Shipped (X,Partner),¬Pay(Partner)⇒ Breach(Pay,Partner)*. In the light of new information such as "there was a delay in money transfer due to the bank", the above rule may be defeated. On the one hand, in case of violation, conform to contract clauses, the victim queries the normative agents for the imposed sanction *S* and he usually activates the next rule: *breach(Pay, Partner)) ⇒ penalty(S,Partner)*. On the other hand, if the violation of a clause is not too serious, or was not intended by the violating party, or the breacher is a regular partner, the contracting parties usually do not want to consider this as a breach of the contract, but simply as a disruption in the execution of the contract that has to be repaired:

r_1 : *regularPartner(X),delay(Short) ⇒ ¬penalty(S,Partner)*

r_2 : *delay(P,medium), regularClient(P)⇒ ¬ breach(P, Contract)*

r_3 : *quality(P, Item, Low),intention(P) ⇒ ¬ breach(P, Contract)*

4.1.2 Context

In order to reason on the rules of the contract, each agent may attach a current context *C(a)*, which depends on the agent point of view: *C=<I, B,C(a)>*. For instance, the rules may be applied differently in case of a contract with a long time business partner or short time one. Also, the relatively importance of the rules depends on the current market conditions. Each agent attaches its own context to the contract. To accommodate this phenomenon, the superiority relation is computed dynamically according to some principles encoded as defeasible rules: *regularPartner* $\Rightarrow r_1 > r_2$.

4.2 Normative agents

Normative agents or expert agents encapsulate defeasible theories derived from the principles of contract law. They can be call by the contractual agents in order to compute a penalty or by the argumentation agent. In this case they have the role of a consultant. They can decide on the validity of a contract or they can apply some doctrines in order to compute the amount of penalty imposed [5] . The normative agents implement the main phases of a legal problem solving: proof of the facts, rule interpretation, and rule application (see fig. 7).

When an agent considers that a breach has occurred, he provides the facts F_1 to the normative agent that was specified in the contract. He notifies the potential breacher, which also provides its facts F_2. The normative agent compares the facts and if the sets do not correspond, the supplemental facts $S_1 = F_1 \setminus F_2$ and $S_2 = F_2 \setminus F_1$ are sent for acceptance to the breacher and victim respectively. Part of the

[5] It is considered that 80% from the cases are simple enough to be resolved automated.

above facts are accepted A_1, while other are rejected R_1 ($A_1 \cup R_1 = S_1$). For the rejected facts, the normative agent asks agents who provided them for proves. The required proves and the information obtained from the monitor agent (third party) are used by the normative agent to decide on the current facts. In the next step, the agent interprets the rules using the above facts, the clauses from the contract, and its own legal defeasible theory. Finally, it computes the imposed remedy.

4.2.1 Contract validity

The goal of this agent is to validate all the clauses which appear in the contract. In law the famous example is "A contract exists because there was offer, acceptance, memorandum, and consideration", which can be defeated by "But the one of the parties to the contract is incompetent" so there is no contract [1]. This agent handles such contract validity issues:

r_1: *offer(S), acceptance(B), memorandum(S,B) consideration(S,B)* $\Rightarrow$ *contract(S, B).*

r_2: *memorandum(S,B), minor(B)* $\sim>$ *contract(S, B)*

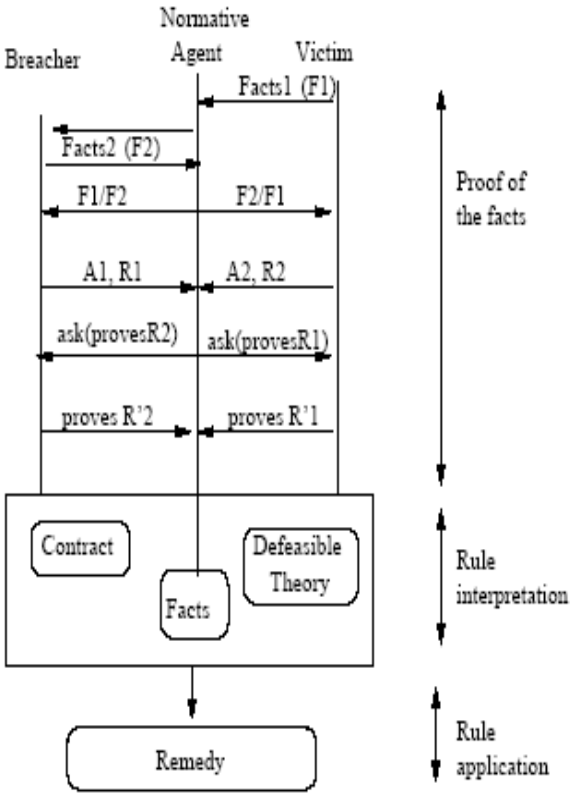

Figure 7. Normative agents.

Consistent with party autonomy, the agents may settle on different remedies at contracting time. This approach increases the flexibility and efficiency, because the agents are the ones who know what type of remedy protect better their interest. Party designed remedies specify themselves the amount of damages in case of breach. Usually, the courts enforce the penalties specified in the contracts, but there are situations when this is not a fair solution. The courts do not enforce a specified penalty that is too exaggerated:

r_1 : *delay(1day)* $\Rightarrow$ *penalty(1000).*

r_2 : *delay(1day)* $\Rightarrow$*) penalty(expectationdamages).*

$r_2 > r_1$.

4.2.2 Expectation damages

According to this agent, the amount of expectation damages must place the victim in the same position as if the actual contract had been performed [2]. The defeasible theory from fig. 8 considers if the agent has signed contracts for all its input items necessary to produce its output, it considers also if the victim has a contract for its output item, meaning that the expected profit would be very probable, and it considers also if the victim has mitigated its losses by taking reasonable actions to reduce losses occurred by the promisor's breach. The market can impose the victim to find substitute items. In this case the imposed damages reflect only the differences between the original contract and the substitute one. This is exactly the kind of situation in which the law's "lex specialis" recommends the more specific rule. In logics of defeasible reasoning the more specific antecedent dominates.

r_1: *bid(Victim,V_p),contracPrice(P_c)* $\Rightarrow$ *expProfit(V_p-P_c)*

r_2:*presumable(Victim),investments(Victim,I_p),outputContract(Victim,R_p)* $\Rightarrow$ *expProfit(R_p-I_p)*

r_3:*presumable(Victim),investments(Victim,I_p),outputContract(Victim,R_p),less(R_p,I_p)* $\Rightarrow$ *expProfit(0)*

r_4: *substitute(P_s),contracPrice(P_c)* $\Rightarrow$ *expProfit(P_c-P_s)*

r_5:*substitute(P_s),contracPrice(P_c),less(P_c,P_s)*$\Rightarrow$ *expProfit(P_c-P_s)*

$r_3 > r_2$, $r_2 > r_1$, $r_5 > r_4$, $r_4 > r_2$

Figure 8. Part from the expectation damages Theory.

r_1: *inputContracts(Victim,0)* $\Rightarrow$ *reliance(0)*
r_2: *inputContracts(Victim,*I_p*), valuation(Victim,*V_p*)* $\Rightarrow$ *reliance(*V_p-I_p*)*
r_3: *inputContracts(Victim,*I_p*), valuation(Victim,*V_p*), presumable(Victim), outputContract(Victim,*R_p*)* $\Rightarrow$ *reliance(*V_p-I_p+R_p*)*
r_4: *substitute(*P_s*), contracPrice(*P_c*)* $\Rightarrow$ *reliance(*P_c-P_s*)*
r_5: *substitute(*P_s*), contracPrice(*P_c*), less(*P_c,P_s*)* $\Rightarrow$ *reliance(*P_c-P_s*)*
r_6: *reliance(*D_r*), contracPrice(*P_c*), less(*P_c,D_r*)* $\Rightarrow$ *remedy(*P_c*)*
r_7: *reliance(*D_r*), contracPrice(*P_c*), less(*P_c,D_r*), notification(*D_r*)* $\Rightarrow$ *remedy(*P_c*)*
$r_2 > r_1$, $r_3 > r_2$, $r_5 > r_4$, $r_4 > r_2$, $r_7 > r_6$

Figure 9. Part from the reliance damages theory.

4.2.3 Reliance damages

According to this agent, the amount of reliance damages must place the victim in the same position as if no contract had been signed [2]. The defeasible theory from fig. 9 considers how many investments the victim made and how optimal these investments were. In some cases damages can be higher than the contract value itself ($D_r > P_c$). According to the current practice in law, these damages could be the right ones if the victim gives a previous notification about the risks faced by the potential breacher.

4.2.4 Opportunity cost

According to this agent, the amount of opportunity cost damages must place the victim in the same position as if the best alternative contract had been performed [2]. First, the normative agent estimates the opportunity cost P_o, which is the transaction cost in case the breacher have been absent from the market. Usually one seller less implies $P_o < P_c$ and one buyer less implies $P_c < P_o$.

4.3 Dispute resolution agents

Each contract specifies in the informative part the remedies that will be imposed. When the partner is not agree with the computed penalty according to the above agents, he can initiate the dispute resolution mechanism by calling the agent which is specified in the corresponding clause from behavioral part of the contract. Therefore, all the above remedies will influence the amount of damages and they are used as arguments when the dispute is arbitrated. Hence, the expected profit, the amount of investments made, the existence of a substitute, the time of breach, the market conditions, all represents a set of different factors, possibly with different magnitudes, that have to be weighed in each particular situation to compute the remedy. Each agent has to consider not only its own knowledge base, but takes into account also the defeasible theory corresponding to the relevant jurisprudence. Arguments may compete rebutting each other, so a process of argumentation is a natural result of the search for arguments. When we chain defeasible rules to reach a conclusion, the agents have arguments, instead of proofs. Adjudication of competing arguments must be performed, comparing arguments in order to determine what beliefs are justified [1].

5. Related Work

Another approach for handling exceptions during the execution of a contract uses DAML OIL for process knowledge description and RuleML for the representation of business rules [5]. Instead of courteous logic we use defeasible logic to deal with conflicted rules. Defeasibility, being one of the main characteristic of the normative reasoning, we consider that defeasible logic is more proper for handling exceptions in contract execution. Moreover, it provides valuable support for argumentation when a dispute arises. The defeasible logic was also used in [3] for contracts monitoring. The authors extend the RuleML to cover deontic and defeasible aspects of legal language. We attack the problem from a more functional point of view by providing agents build in OAA framework responsible for computing the remedies and for providing arguments in case of a dispute resolution.

The task dependency network model was proposed [11] as an efficient market mechanism in achieving supply chain coordination. The authors analyze protocols for agents to reallocate tasks for which they have no acquired rights. However, this approach is rather a timeless-riskless economy. Moreover, the breach

of a contract implies no penalties, which is an unrealistic assumption in real world. In contrast, in our model we have introduced penalties in case of contract breaching and we have used the task dependency network model as a framework for information extracting and contracts monitoring.

In [10] the author investigates also the automation of the contractual relationships. In that model, the contractual agents consider both the norm and the attached sanction when decide to breach the contract or not. In real contracts, due to the market nondeterminism or to the cost of contract negotiation, not all the norms have a corresponding penalty. In out work we capture these aspects by introducing the possibility to specify in the contract the legal principle (expectation damages, reliance damages, opportunity costs), which will be applied in case of breach. Moreover, we provide also agents responsible for handling situations when a dispute arises.

6. Conclusions

We describe ongoing work on both well-known general priority rules such as the principles of specificity (Lex Specialis), temporality (Lex Posterior), and hierarchy (Lex Superior). Also domain dependent priority rules have been considered. The contributions of this paper consist in providing a framework in which different types of contract exceptions can be handled with defeasible logic and also in introducing penalties in the task dependency network model.

Our current goal is to analyze how the exceptions or the breaches of the contracts are propagated within this task dependency network and how defeasible logic deals with such scalability issues. On the other hand, we can use the task dependency network in order to identify the remedy that would produce the most efficient result, if that remedy were allowed to govern the parties as a default one. For instance, if the reliance damages were determined to be the most efficient penalty for breach of contract, this approach to selecting a default rule would argue for making reliance damages the default remedy.

References

[1] Boella, G. and van der Torre, L. (2004) *Contracts as legal institutions in organizations of autonomous agents*, In AAMAS'04, pages 948-955.

[2] Craswell, R. (2000) Contract law: *General theories*. In B. Bouckaert and G. D. Geest, editors, Encyclopedia of Law and Economics, Volume III. The Regulation of Contracts, pages 1-24. Cheltenham.

[3] Governatori, G (2005) *Representing business contracts in RuleML*. Journal of Cooperative Information Systems, 14(5).

[4] Governatori, G., Maher, M., Antoniou G., and Billington, D. (2000) *Argumentation semantics for defeasible logics.* In Proceedings of the Pacific Rim International Conference on AI.

[5] Grosof, B. N. and Poon, T., (2002). *Representing agent contracts with exception using XML rules, ontologies, and process descriptions.* In International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, Sardinia, Italy.

[6] Maher, M. (2001) *Propositional defeasible logic has linear complexity*. Theory and Practice of Logic Programming, 1(6):691-711.

[7] Maher, M., Rock, A., Antoniou, G., Billington, D. and Miller, T. (2001) *Efficient defeasible reasoning systems,* International Journal of Artificial Intelligence Tools, 10(4):483-501.

[8] McCarthy, J. (1997) *Modality, si! modal logic, no!* Studia Logica, 59.

[9] Milosevic, Z., Gibson, S., Linington, P. F., Cole, J. and Kulkarni, S. (2004) *On design and implementation of a contract monitoring facility*. In First IEEE International Workshop on Electronic Contracts, pages 62-70.

[10] Salle, M. (2002) *Electronic contract framework for contractual agents*. In Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, pages 349-353.

[11] Walsh, M. and Wellman, E.(2003) *Decentralized supply chain formation: A market protocol and competitive equilibrium analysis.* Journal of Artificial Intelligence Research, 19: 513-567