A Simple Argumentation Based Contract Enforcement Mechanism

Nir Oren and Alun Preece and Timothy J. Norman

Department of Computing Science, University of Aberdeen, Aberdeen, AB24 3UE, Scotland
noren,apreece,tnorman@csd.abdn.ac.uk

Abstract. Agents may choose to ignore contract violations if the costs of enforcing the contract exceed the compensation they would receive. In this paper we provide an argumentation based framework for agents to both decide whether to enforce a contract, and to undertake contract enforcement actions. The framework centers around the agent reasoning about what arguments to put forth based on a comparison between the utility it would gain for proving its case and the utility it loses for probing environment state.

1 Introduction

Open environments may contain self–interested agents with different levels of trustworthiness. While self–interested, these agents may both cooperate and compete so as to increase their own utility. Many mechanisms have been proposed to ensure correct agent behaviour in such environments, and most make use of some form of implicit or explicit contract between the agents[1, 17, 6]. The purpose of such a contract is to lay out what is expected from each contracting party. Given norm-autonomous agents, i.e. agents which are able to decide whether to fulfil their normative requirements, contracts also allow for the imposition of penalties and compensation to the wronged party if any deviations from the agreed upon behaviour occurs. Sanctioning of agents often takes place through the use of a trust or reputation framework[11], or some monetary mechanism.

In the real world, minor contract violations are often ignored, either due to the loss in trust that would arise between the contracting parties, or due to the small compensation the wronged party would receive when compared to the overhead of enforcing the contract. Even major violations might not result in the wronged party being (fully) compensated, or the guilty party being penalised as the cost of proving the violation might exceed the compensation which would have been obtained by the victim, resulting in them not attempting to enforce the contract. While the former behaviour might be useful to replicate within multi-agent systems (due to increased efficiency), at first glance the latter behaviour seems undesirable. Such behaviour is however rational (and thus desirable in many settings), as it maximises an agent's gain. It could be argued that loss making contract enforcement actions, which might increase the society's welfare as a whole, are the responsibility of some "pro-bono" third party agents, rather than contract participants.

Contract enforcement costs are not constant in many scenarios. Referring again to a typical real world example, if a contract case goes to court, extra costs are incurred due not only to lawyer's fees, but also due to the cost of gathering evidence. As the case progresses, additional evidence might be needed, leading to further escalating costs. Some legal systems avoid this by having the loser of a case pay its opponent's fees.

The increasing complexity of artificial agent environments means that many of these scenarios have analogies within the agent domain. Agents interacting with each other on the web, virtual marketplace or a Grid do not trust each other and sign contracts before providing and consuming services. If one agent believes another did not fulfil its obligations, it may need to verify its belief by gathering large amounts of evidence. This evidence gathering might cost it not only computational, but also monetary resources as it might have to purchase information from other agents. In a similar manner, it might cost the accused agent resources to defend itself. Allowing for such behaviour can increase both the efficiency and robustness of agent environments.

In this paper we examine multiple issues related to this type of contract enforcement. We provide an argumentation/dialogue game based framework which allows agents to both decide and undertake contract enforcement actions. We also look at how aspects of this framework can tie into contracting languages. Our work forms part of the CONOISE-G project [12]. CONOISE-G centers around the creation and implementation of technologies designed to improve the performance and robustness of virtual organisations. Agents operating within the environment have their behaviour regulated by contracts, and contract monitoring and enforcement thus form a major part of the project focus.

When an agent believes that a contract it is participating in has been violated, it calculates the amount of utility it would (lose) gain by (not) enforcing the contract. While a net utility gain exists, the agent maintains its enforcement action, bringing up evidence as required. The action of presenting evidence decreases the agent's utility. The accused agent follows a similar process, computing how much utility it would lose by not defending itself, and paying for evidence it uses in its defence. This process ends when one agent either capitulates or has no further evidence to present, after which a decision regarding the status of the contract can be reached. While simple, we believe that this approach can both be useful in a large number of scenarios as well as provide the basis for more complicated techniques.

In the next section we formalise our framework, after which an small example is presented. Section 4 looks at the features of our framework, and places it within the context of related work. Finally, possible extensions to this work are discussed.

2 The Formalism

In this section, we describe our approach in detail. We are primarily interested in only one section of the contract enforcement stage, namely the point at which an agent attempts to prove that another agent has (or has not) broken a contract. Informally, the agent begins by determining how much utility it would gain by proving that it has been wronged, as well as what the net utility gain would be for not being able to prove its claims. A dialogue then begins between the accuser and the accused. In the course of this dialogue, evidence is presented from outside sources. Presenting this evidence costs, imposing an ordering on the best way to present the evidence, as well as possibly causing an agent to give up on its claims. Once the agents have made all the utterances they desire, an adjudication process can take place, determining whether an agent has been able to prove its case. The work presented here is an extension of the work described in [9, 8].

We begin by describing the logical layer in which interaction takes place, and the way arguments interact with each other. We decided against using an abstract argumentation framework (such as the one described by Dung [3]) or a legal based argumentation framework (such as Prakken and Sartor's [15]) as our arguments are grounded and do not make use of any default constructs. Making use of our own logical formalism also helps simplify the framework.

After describing the logical level, we specify the dialogue game agents can use to perform contract monitoring actions, examining strategies agents can use to play the game, as well as looking at how to determine the winners and losers of an instance of the game. It should be noted that we discuss very few of our design decisions in this section, instead simply presenting the framework. An in depth examination of the framework is left for Section 4. The section concludes by describing how to transform a contract into a form usable by the framework.

2.1 The Argumentation Framework

Argumentation takes place over the language Σ, which contains propositional literals and their negation.

Definition 1. ***Argument.*** *An argument is a pair (P, c), where $P \subseteq \Sigma \cup \{\top\}$ and $c \in \Sigma$ such that if $x \in P$ then $\neg x \notin P$. We define $Args(\Sigma)$ to be the set of all possible arguments derivable from our language.*

P represents the premises of an argument (also referred to as an argument's support), while c stands for an argument's conclusion. Informally, we can read an argument as stating "if the conjunction of its premises holds, the conclusion holds". An argument of the form $(\top, a)$ represents a conclusion requiring no premises (for reasons detailed below, such an argument is not necessarily a fact).

Arguments interact by supporting and attacking each other. Informally, when an argument attacks another, it renders the latter's conclusions invalid.

An argument cannot be introduced into a conversation unless it is grounded. In other words, the argument $(\{a, b\}, c)$ cannot be used unless a and b are either known or can be derived from arguments derivable from known literals. Care must be taken when formally defining the concept of a grounded argument, and before doing so, we must (informally) describe the proof theory used to determine which literals and arguments are justified at any time.

To determine what arguments and literals hold at any one time, let us assume that all arguments refer to beliefs. In this case, we begin by examining grounded beliefs and determining what can be derived from them by following chains of argument. Whenever a conflict occurs (i.e. we are able to derive literals of the form x and $\neg x$), we remove these literals from our derived set. Care must then be taken to eliminate any arguments derived from conflicting literals. To do this, we keep track of the conflicting literals in a separate set, and whenever a new conflict arises, we begin the derivation process afresh, never adding any arguments to the derived set if their conclusions are in the conflict set.

Differentiating between beliefs and facts makes this process slightly more complicated. A literal now has a chance of being removed from the conflict set if it is in the set of known facts.

More formally, an instance of the framework creates two sets $J \subseteq Args(\Sigma)$ and $C \subseteq \Sigma$, while making use of a set of facts $F \subset \Sigma$ such that if $l \in F$ then $\neg l \notin F$ and if $\neg l \in F$ then $l \notin F$ (i.e. F is a consistent set of literals). J and C represent justified arguments and conflicts respectively.

Definition 2. ***Derivation.*** *An argument $A = (P_a, c_a)$ is derivable from a set S given a conflict set C (written $S, C \vdash A$) iff $c_a \notin C$ and ($\forall p \in P_a(\exists s \in S$ such that $s = (P_s, p)$ and $p \notin C)$ or $P_a = \{\top\}$).*

Clearly, we need to know what elements are in C. Given the consistent set of facts F and a knowledge base of arguments $\kappa \subseteq Args(\Sigma)$[1] , this can be done with the following reasoning procedure:

$$J_0 = \{A | A \in \kappa \text{ such that } \{\}, \{\} \vdash A\}$$
$$C_0 = \{\}$$

Then, for $i > 0, j = 1 \dots i$, we have:

$$C_i^* = C_{i-1} \cup \{c_A, \neg c_A | \exists A = (P_A, c_A), B = (P_B, \neg c_A) \in J_{i-1}\}$$
$$C_i = C_i^* \backslash (C_i^* \cap F)$$

$$X_{i0} = \{A | A \in \kappa \text{ and } \{\}, C_i \vdash A\}$$
$$X_{ij} = \{A | A \in \kappa \text{ and } X_{i(j-1)}, C_i \vdash A\}$$

$$J_i = X_{ii}$$

The set X allows us to recompute all derivable arguments from scratch after every increment of i[2]. Since i represents the length of a chain of arguments, when $i = j$ our set will be consistent to the depth of our reasoning, and we may assign all of these arguments to J. Eventually, $J_i = J_{i-1}$ (and $C_i = C_{i-1}$) which means there are no further arguments to find. We can thus define the conclusions reached by a knowledge base κ as $K = \{c | A = (P, c) \in J_i\}$, for the smallest i such that $J_i = J_{i+1}$. We will use the shorthand $K(\kappa)$ and $C(\kappa)$ to represent those literals which are respectively derivable from, or in conflict with a knowledge base κ. C_i^* represents the conflict set before facts are taken into account.

[1] We assume that κ contains all our facts, i.e. $\forall f \in F, f \in \kappa$

[2] This allows us to get rid of long invalid chains of arguments, as well as detect and eliminate arbitrary loops.

2.2 The Dialogue Game

Agents make use of the argumentation framework described above in an attempt to convince others of their point of view. An agent has an associated private knowledge base (KB) containing its beliefs, as well as a table listing the costs involved in probing the system for the value of literals (M). An instance of the argumentation dialogue is centred around agents trying to prove or disprove a set of goals G. Utility gains and losses are associated with succeeding or failing to prove these goals. The environment also contains a public knowledge base recording the utterances made by the agents. This knowledge base performs a role similar to a global commitment store, and is thus referred to as CS below.

Definition 3. ***Environment.*** *An environment is a tuple $(Agents, CS, F, S)$ where Agents is the set of agents participating in the dialogue, $CS \subseteq Args(\Sigma)$ is a public knowledge base and $F \subset \Sigma$ is a consistent set of literals known to be facts. $S \subseteq \Sigma$ contains literals representing the environment state.*

Definition 4. ***Agent.*** *An agent $\alpha \in Agents$ is composed of a tuple $(Name, KB, M, G, U_{win}, U_{draw}, U_{lose}, T)$ where $KB \subseteq Args(\Sigma)$, $G \subseteq \Sigma$, M is a function allowing us to compute the cost of probing the value of a literal. U_{win}, U_{draw}, $U_{lose} \in \mathbb{R}$ are the utilities gained for winning, drawing or losing an argument. $T \in \mathbb{R}$ keeps track of the total costs incurred by an agent during the course of the argument.*

The monitoring cost function M expresses the cost incurred by an agent when it must probe the environment for the value of a literal. It maps a set of literals to a real number:

Definition 5. ***Monitoring costs.*** *The monitoring cost function M is a domain dependant function $M : 2^{\Sigma} \rightarrow \mathbb{R}$*

Representing monitoring costs in this way allows us to discount multiple probing actions, for example, it might be cheaper for an agent to simultaneously determine the cost of two literals than to probe them individually in turn.

Agents take turns to put forward a line of argument and ascertain the value of a literal by probing the environment. For example $\{((\top, a), (a, b)), b)\}$ is a possible utterance an agent could make, containing the line of argument $\{(\top, a), (a, b)\}$ and probing the environment for whether b is indeed in the environment state. Alternatively, an agent may pass by making an empty utterance $\{,\}$. The dialogue ends when CS has remained unchanged for as many turns as there are players, i.e. after all players have had a chance to make an utterance, but didn't. Once this has happened, it is possible to compute the literals derivable from CS, determine the status of an agent's goal expression, and thus compute who won the dialogue.

Definition 6. ***Utterances.*** *The utterance function*

$$utterance : Environment \times Name \rightarrow 2^{Args(\Sigma)} \times \Sigma$$

accepts an environment and an agent name, returns the utterance made by the agent. The first part of this utterance lists the arguments advanced by the agents, while the second lists the probed environment states.

Given an agent with a monitoring cost function M, we may compute the cost to the agent of making the utterance (Ar, Pr), where Ar is the line of argument advanced by the agent and Pr is the set of literals the agents would like to probe, as $M(Pr)$.

Definition 7. ***Turns.*** *The function*

$$turn : Environment \times Name \rightarrow Environment$$

takes an environment and an agent label, and returns a new environment containing the effects of the agent's utterance.

Given an environment $Env = (Agents, CS, F, S)$ *and an agent* $\alpha = (Name, KB, M, G, U_{win}, U_{draw}, U_{lose}, T) \in Agents$*, we define the turn function as follows*

$$turn(Env, Name) = (NewAgents, CS \cup Ar, F \cup (Pr \cap S), S) \text{ where } Ar, Pr$$
$$\text{are computed from the function } utterance(Env, Name) = (Ar, Pr), \text{ and}$$
$$NewAgents = Agents \backslash \alpha \cup (Name, KB, M, G, U_{win}, U_{draw}, U_{lose}, T + M(Pr))$$

We may assume that the agents are named $Agent_0, Agent_1, \ldots, Agent_{n-1}$ where n is the number of agents participating in the dialogue. It should be noted that the inner workings of the $utterance$ function are dependant on agent strategy, and we will describe one possible game playing strategy below. Before doing so however, we must define the dialogue game itself. Each turn of the dialogue game results in a new environment, which is used during later turns.

Definition 8. ***Dialogue game.*** *The dialogue game can be defined in terms of the turn function as follows:*

$$turn_0 = turn((Agents, CS_0, F_0, S), Agent_0)$$
$$turn_{i+1} = turn(turn_i, Agent_{i \text{ mod } n})$$

The game ends when $turn_i \ldots turn_{i-n+1} = turn_{i-n}$.

CS_0 and F_0 contain the initial arguments and facts, and are usually empty. Note that the agent may make a null utterance $\{,\}$ during its move to (eventually) bring the game to an end.

To conclude the dialogue game definition, we need to determine how much utility an agent gains at the end of a dialogue instance. An agent wins the game if it is able to prove all its goals. A draw occurs when the status of an agent's goals are unknown (either due to being indeterminable or in the conflict set).

Definition 9. ***Agent utility.*** *Given an* $environment = (Agents, CS, F, S)$*, and abbreviating an agent definition* $(Name, KB, M, G, U_{win}, U_{draw}, U_{lose}, T)$ *as* α*, the winning set of agents is defined as*

$$Agent_{win} = \{\alpha | \alpha \in Agents \text{ such that } G \subseteq K(CS)\}$$

The set of drawing agents is then defined as

$$Agent_{draw} = \{\alpha | \alpha \in Agents \textit{ such that}$$
$$\forall g \in G, (g \in C(CS) \textit{ or } \{g, \neg g\} \cap K(CS) = \{\})\}$$

All other agents are in the losing set: $Agent_{lose} = Agents \backslash (Agent_{win} \cup Agent_{draw})$.

An agent α *may calculate its utility in such an environment by computing*

$$U(\alpha) = \begin{cases} U_{win} - T & \textit{if } \alpha \in Agent_{win} \\ U_{draw} - T & \textit{if } \alpha \in Agent_{draw} \\ U_{lose} - T & \textit{if } \alpha \in Agent_{lose} \end{cases}$$

Note that drawing (or even losing) a dialogue game may provide an agent with more utility than winning the game. At the end of the game, $K(CS)$ contains all literals which are agreed to be in force by the agents, while $C(CS)$ contains all conflicting literals.

2.3 The Heuristic

We are now in a position to define one possible technique for taking part in the dialogue game. We assume that our agent is rational and will thus attempt to maximise its utility. By using the reasoning procedure described in Section 2.1 over the environment's knowledge base CS, its knowledge base KB and the set of known facts F, an agent can both determine what literals are currently in force and in conflict, as well as determine the effects of its arguments. Given all possible arguments to advance $PA \in 2^{KB}$, we can define the resultant commitment store by computing $RCS = PA \cup CS$. If we call PF the set of possible facts which the agent an probe, we can compute the set of possible utterances as $PU = RCS \times PF$. By then performing the move for each PU and computing the agent's utility as if the game had ended, an ordering based on utility of the elements of PU can be generated. The agent then returns the element of PU which maximises its utility.

Given a set of possible utterances with equal utility, we use a secondary heuristic (as described in [9]) to choose between them: the agent will make the utterance which reveals as little new information to the opponent as possible. More formally,

Definition 10. ***Making utterances.*** *For an environment* Env *and an agent* $\alpha = (Name, KB, M, G, U_{win}, U_{draw}, U_{lose}, T)$*, let* $PA \in 2^{KB}$*,* $RCS = PA \cup CS$*. We compute a set of possible facts accessible by the agent (*PF*) as*[3]

$$PF = \{f, \neg f | f \in (K(RCS) \cup C(RCS)) \backslash F\} \textit{ and } \{f, \neg f\} \cap S \neq \{\}$$

The set of all possible utterances is thus $PU = RCS \times PF$*. Let* $Env_{new}(Utt) = turn(Env, Name)$ *where* Utt *is the utterance function used within the turn function. We can compute the utility of* Utt *as* $U_{utterance}(Utt) = U(\alpha, Env_{new})$*.*

Then the agent will make the utterance $Utt = (Ar, Pr) \in PU$ *such that* $max_{Utt \in PU}(U_{utterance}(Utt))$*. If multiple such possible utterances exist, we will choose one such that* $K(Ar \cup CS) - K(CS) + C(Ar \cup CS) - C(CS)$ *is minimised.*

[3] The second part of the condition allows us a way of limiting the probing to only those facts which are in fact accessible, without having to know their value

Assuming that every probing action has an associated utility cost, such an agent would begin by attempting to argue from its beliefs, probing the environment only as a last resort. This behaviour is reminiscent of the idea put forward by Gordon's pleadings game [4], where agents argue until certain irreconcilable differences arise, after which they approach an arbitrator to settle the matter.

It should also be noted that our framework allows for different probing costs to be associated with probing for a value or its negation. This makes sense in a contracting environment, as different sensors might be used to perform these two types of probing actions.

2.4 Contracts

To utilise such a framework in the context of contracting requires a number of additional features:

1. S, the set of facts which can be probed, must be defined.
2. T the agent's cost for performing the probing must also be determined.
3. G the set of agent goals must be computed.
4. The agent's winning, drawing and losing utilities must be set appropriately.
5. The agent's knowledge bases KB must be created to reflect the content of the contract, as well as any initial beliefs held by the agents regarding the environment state.
6. F_0, the set of known facts must be generated.

While all of these are contract specific, some guidelines can be provided due to the features of the framework. Given a two party contract, we can assign our agents plaintiff and defendant roles. Usually, they will have opposite goals, initially determined by a combination of the contract clauses and the the plaintiff's beliefs regarding the environment state. The set S, as well as the cost of performing the probing is determined by the contract, and the contract clauses, together with an agent's beliefs about the state of the world are used to determine an agent's KB. F_0 (and thus CS_0) will not be empty if certain facts about the environment state are already known.

Item 4 is interesting. Most legal systems operate under the requirement that a plaintiff prove their case either on the balance of probabilities, or beyond reasonable doubt. Given the binary nature of our framework, both reduce to the same level. This means that the winning and drawing utilities for the defendant will be the same, while the drawing and losing utilities of the plaintiff will be identical. For many contracts, The winning utility of the plaintiff will be the same as the losing utility of the defendant (reflecting the fact that the defendant will have to pay the plaintiff in the case of a loss).

To simplify matters, we assume that a contract is enforced in its entirety, i.e. all issues must be settled in favour of the plaintiff for them to win. We thus define a contract as follows:

Definition 11. ***Contract.*** *A contract consists of the tuple* $(\mathit{Plaintiff}, \mathit{Defendant}, Clauses, Monitors, Issues, Penalty)$ *where* $\mathit{Plaintiff}$ *and* $\mathit{Defendant}$ *are labels,* $Clauses \in Args(\Sigma)$*,* $Monitors : \{\mathit{Plaintiff}, \mathit{Defendant}\} \times 2^{\sigma} \rightarrow \mathbb{R}$*,* $Issues \subseteq \Sigma$*, and* $Penalty \in \mathbb{R}$*.*

Given such a contract, as well as a set of states S, we can instantiate our framework as follows:

$$Environment = ((Agents_0, Agents_1),,, S)$$

$$Agents_0 = (\mathit{Plaintiff}, Clauses, M_{\mathit{Plaintiff}}, Issues, Penalty, 0, 0, 0)$$

$$Agents_1 = (Defendant, Clauses, M_{Defendant}, Issues, 0, Penalty, Penalty, 0)$$

Where $M_{\mathit{Plaintiff}}$ and $M_{Defendant}$ are computed by partially parameterising the $Monitors$ function with the appropriate label. Note that the defendant's goals are the same as the plaintiff's goals, but that it gains utility for "losing" or drawing the game, as this would mean it had successfully defended it's stance.

At this stage, contract enforcement is possible using the framework. We will now provide a short example to illustrate the framework in operation.

3 Example

We will look at a very simple scenario (taken from the service provision scenario described in [12]) where a provider agent has agreed to provide a movie service to a consumer agent, subject to restrictions on the movie framerate.

Given the following contract clauses

$$\begin{aligned}
&fr25 \rightarrow payPerson \\
&\neg fr25 \rightarrow giveWarning1 \\
&wrongMovie \rightarrow giveWarning2 \\
&giveWarning1 \wedge giveWarning2 \rightarrow penalty
\end{aligned}$$

We assume that monitors exist for $fr25, giveWarning1$ and $giveWarning2$ at a cost of 5,10 and 20 respectively. Finally, let the penalty for contract violation be 30 units of currency.

Now let us assume that the consumer believes that it has been given the incorrect movie, and when the movie finally arrived, its framerate was below 25 frames per second (i.e. the literal $\neg fr25$ evaluates to true). Furthermore, the provider disputes all of this, believing that it provided the right movie at an appropriate framerate. After creating the agents using the method described in Section 2.4, the following conversation might take place (omitting brackets for the sake of readability where necessary):

$$\begin{aligned}
&(P1)\ (\{(\neg fr25, giveWarning1), (wrongMovie, giveWarning2), \\
&\qquad (\{giveWarning1, giveWarning2\}, penalty)\}, \{\}) \\
&(D2)\ ((\top, fr25), \{\}) \\
&(P3)\ (\{\}, \{\neg fr25, fr25\}) \\
&(D4)\ ((\top, \neg wrongMovie) \\
&(P5)\ (\{\}, \{\neg giveWarning2, giveWarning2\}) \\
&(D6)\ () \\
&(P7)\ ()
\end{aligned}$$

The plaintiff first puts forward its case, based on its beliefs. Since the agent attempts to reveal as little as possible, the defendant utters just enough to counter the plaintiff's argument. The plaintiff responds by giving proof for its argument, as that is all it can do. Note that the state of $fr25$ rather than $giveWarning1$ was probed due to its lower utility cost. This process repeats itself for $wrongMovie$, but since this literal is not directly observable, the agent must probe its conclusion instead. Finally, no more arguments are put forward, and the case is decided in favour of the plaintiff, who earns a net utility of 5.

Had the defendant attempted to argue for its beliefs regarding the state of $fr25$ in an earlier contract enforcement episode, then this round of argument may have begun with $\neg fr25$ already being an established fact (i.e. part of F_0). As can be seen it is difficult to provide an all encompassing domain independent set of rules to convert a contract, agents, and environment into a form suitable for a contract enforcement action.

4 Discussion

While we have focused on using our framework for contract enforcement, it can also be used in other settings. For example, given a non-adversarial setting where probing sensors still has some associated cost (for example, of network resources or time), an agent can reason with the framework (by generating an argument leading to its goals) to minimise these sensing costs.

The contract enforcement stage is only part of the greater contracting life-cycle. With some adaptation, our framework can also be used in the contract monitoring stage: by constantly modifying its beliefs based on inputs from the environment, an agent could continuously attempt to prove that a contract has failed; once this occurs contract enforcement would begin.

Contract enforcement and monitoring has been examined by a number of other researchers. Given a fully observable environment in which state determination is not associated with a utility cost, the problem reduces to data mining. Research such as [19] operates in such an environment, but focus more on the problem of predicting imminent contract failure. Daskalopulu et al. [2] have suggested a subjective logic [5] based approach for contract enforcement in partially observable environments. Here, a contract is represented as a finite state machine, with an agent's actions leading to state transitions. A central monitor assigns different agents different levels of trust, and combines reports from them to determine the most likely state of the system. While some weaknesses exist with this approach, most techniques for contract enforcement are similar in nature, making use of some uncertainty framework to determine what the most likely system state is, then translating this state into a contract state, finally determining whether a violation occurred. An argumentation based approach potentially has both computational as well as representational advantages over existing methods. In earlier work[10], we described a contracting language for service level agreements based on semantic web standards (called SWCL). One interesting feature of that work is the appearance of an explicit monitoring clause describing where to gather information regarding specific environment states. Most other contracting languages lack such a feature, and the addition of a monitoring cost would allow SWCL to be used as part of our framework. A related feature of our framework which, in a contracting context would require a language with appropriate capabilities, is the ability to assign different monitoring costs for determining whether a literal or its negation holds. In an open world environment, such a feature is highly desirable.

Argumentation researchers have long known that a dialogue should remain relevant to the topic under discussion [7]. This trait allows dialogue based systems to rapidly reach a solution. The approach presented here enforces this requirement due to the nature of the heuristic; any extraneous utterances will lead to a reduction in an agent's final utility. One disadvantage of our approach is that, as presented, the computational complexity of deciding what utterance to make is exponential in nature. Simple optimisations can be implemented to reduce the average case complexity, but in the worst case, all possible arguments must still be considered. Mitigating this is the fact that the number of clauses involved in a contract enforcement action is normally relatively small, making its use practical in the contracting domain.

Many different argumentation frameworks have been proposed in the literature ([16] provides an excellent overview of the field). We decided to design our own framework rather than use an existing approach for a number of reasons. First, many frameworks are abstract in nature, requiring the embedding of a logic, and then making use of some form of attacking relation to compute which arguments are, or are not in force. Less abstract frameworks focus on the non–monotonic nature of argument, often requiring a default logic be used. The manner in which agents reason using our heuristic, as well as the grounded nature of the subject of arguments in our domain makes the argumentation framework presented here more suitable than others for this type of work. However, we intend to show the relationship between our framework and sceptical semantics in existing argumentation frameworks in future work.

Legal argumentation systems often grapple with the concept of burden of proof (e.g. [13, 14, 18]). We attempt to circumnavigate the problem of assigning responsibility for proving the state of a literal to a specific agent by having agents probe for the value themselves as needed. This approach will not work in more complicated scenarios with conflicting sensors, and extending the framework to operate in such environments should prove interesting.

One quirk of our framework is that we do not do belief revision when agents are presented with facts. While adapting the method in which $NewAgents$ are created in Definition 7 is possible by setting the new agent's KB to be $KB \cup (\top, f) \forall f \in F$, and even remove any "obviously conflicting" beliefs, we are still unable to remove beliefs that arise from the application of chains of arguments. We would thus claim that an agent's beliefs are actually a combination of its private knowledge base KB, the public knowledge base CS and the set of presented facts F, rather than being solely a product of KB. Overriding beliefs with facts means our framework assigns a higher priority to fact based argument than belief based argument. This is reminiscent of many existing priority based argumentation frameworks such as [15].

Another possible area of future work involves reasoning about contracts with multiple weakly related clauses. Currently, an agent wins or loses an argument based on whether it can prove all its goals. This (unrealistic) assumption simplifies the problem greatly. By enriching the framework with a more complicated reward function, an agent would be able to gain (or lose) utility by proving only some of its goals. Such work would probably need other enhancements such as opponent modelling and the integration of a planner to allow the agents to plan arguments further than just its next utterance.

Finally, the procedure used to transform a contract into an environment and agents for argumentation is very simple. Enhancing this procedure to make use of the full power of the argumentation framework requires further examination. This enhancement will allow for both the representation of, and dialogue regarding, more complex contracts, further increasing the utility of the framework. Another area of future work involves n–party contracts. While our framework provides support for such dialogue, agents, we have not examined what such contracts would look like, and this might be an interesting research direction to pursue.

5 Conclusions

Explicit or implicit contracts are the dominant method for specifying desired agent behaviour within complex multi-agent systems. Contract enforcement is necessary when agents are able to renege on their obligations.

In this paper we have presented an argumentation based framework for contract enforcement within partially observable environments for which querying sensors has an associated cost. This work can prove useful in a variety of settings, including untrusted (and trusted) distributed computing environments such as the Grid. While many interesting research questions remain, we believe that our framework provides a good starting point to model, and reason about such environments.

Acknowledgements

This work is partly funded by the DTI/EPSRC E-Science Core Program and BT, via a grant for the CONOISE-G project (http://www.conoise.org), a multi-university collaboration between Aberdeen, Cardiff and Southampton universities, and BT.

References

1. R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational–mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.
2. A. Daskalopulu, T. Dimitrakos, and T. Maibaum. Evidence-based electronic contract performance monitoring. *Group Decision and Negotiation*, 11(6):469–485, 2002.
3. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.
4. T. F. Gordon. The pleadings game: formalizing procedural justice. In *Proceedings of the fourth international conference on Artificial intelligence and law*, pages 10–19. ACM Press, 1993.
5. A. Josang. Subjective evidential reasoning. In *Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1671–1678, July 2002.

6. M. J. Kollingbaum and T. J. Norman. Supervised interaction – creating a web of trust for contracting agents in electronic environments. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi–Agent Systems*, pages 272–279, 2002.
7. D. Moore. *Dialogue game theory for intelligent tutoring systems*. PhD thesis, Leeds Metropolitan University, 1993.
8. N. Oren, T. J. Norman, and A. Preece. Arguing with confidential information. In *Proceedings of the 18th European Conference on Artificial Intelligence*, page To appear, Riva del Garda, Italy, August 2006.
9. N. Oren, T. J. Norman, and A. Preece. Loose lips sink ships: a heuristic for argumentation. In *Proceedings of the Third International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2006)*, page To Appear, Hakodate, Japan, May 2006.
10. N. Oren, A. D. Preece, and T. J. Norman. Service level agreements for semantic web agents. In *Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web*, 2005.
11. J. Patel, W. Teacy, N. Jennings, and M. Luck. A probabilistic trust model for handling inaccurate reputation sources. In *Proceedings of Third International Conference on Trust Management*, pages 193–209, 2005.
12. J. Patel, W. T. L. Teacy, N. R. Jennings, M. Luck, S. Chalmers, N. Oren, T. J. Norman, A. Preece, P. M. D. Gray, Shercliff, P. J. G., Stockreisser, J. Shao, W. A. Gray, N. J. Fiddian, and S. Thompson. Agent-based virtual organisations for the grid. *International Journal of Multi-Agent and Grid Systems*, 1(4):237–249, 2005.
13. H. Prakken. Modelling defeasibility in law: Logic or procedure? *Fundamenta Informaticae*, 48(2-3):253–271, 2001.
14. H. Prakken, C. A. Reed, and D. N. Walton. Argumentation schemes and burden of proof. In *Workshop Notes of the Fourth Workshop on Computational Models of Natural Argument*, 2004.
15. H. Prakken and G. Sartor. A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law*, 4:331–368, 1996.
16. H. Prakken and G. Vreeswijk. Logics for defeasible argumentation. In D. Gabbay and F. Guenthner, editors, *Handbook of philosophical logic, 2nd Edition*, volume 4, pages 218–319. Kluwer Academic Publishers, 2002.
17. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1–2):231–252, 1995.
18. D. N. Walton. Burden of proof. *Argumentation*, 2:233–254, 1988.
19. L. Xu and M. A. Jeusfeld. *Pro-active monitoring of electronic contracts*, volume 2681 of *Lecture notes in Computer Science*, pages 584–600. Springer-Verlag GmbH, 2003.