A Challenge for Multi-Party Decision Making: Malicious Argumentation Strategies

Andrew Kuipers and **Jörg Denzinger**
Department of Computer Science
University of Calgary, Canada

Abstract

We present the concept of malicious argumentation strategies that extends malicious argumentation tactics to manipulate the outcome of an argumentation based decision making process with resource limits. We give an example of such a strategy, Exhaust and Protract, and show in a decision making example how Exhaust and Protract can be used to change the result of the decision making process.

Introduction

There are many potential problems automated decision making faces. Among these problems is the potential inability to agree on the goals the decision making has to fulfill or how to rank given goals. This problem is nearly guaranteed to appear if the decision making requires several parties to agree on a decision. The use of argumentation is considered as a solution to this problem. The involved parties use some kind of agreed scheme regulating what arguments are and how an argument can be countered by other arguments and after the scheme is played through the argument(s) "still standing" determine the decision.

Formally, the scheme is represented as a so-called argumentation game (McBurney and Parsons 2002), arguments are realized as formulae in a logic and whether an argument counters another is checked using some (automated) deduction method for the logic. Since usually decisions have to be made within a certain time limit, this means that we cannot assume that the deduction process (or processes) has finished its work, which means that some default outcome needs to be introduced. Unfortunately, such a default opens the door to what is called *malicious argumentation* (Kuipers and Denzinger 2010) which aims at creating arguments that are guaranteed to bring the deduction process to the default outcome.

(Kuipers and Denzinger 2010) presented several ways how a single argument can be rewritten so that checking it with limited resources against a given set of knowledge does not reveal a contradiction although with no limitations a contradiction can be found. While for very specialized argumentation games with the "right" default this is already enough, more complex games cannot be won with such a tactic alone. In fact, games involving a dialog between the arguing parties require what we call *malicious argumentation strategies* to be taken advantage off by a malicious party. Such strategies combine the use of various tactics in different steps of the game.

In this paper, we present a collection of tactics that can be combined by a malicious strategy and then present one strategy, called *Exhaust and Protract* that aims at winning an argumentation game by using first a tactic creating an argument that cannot be countered by the opponent due to resource exhaustion followed by arguments created by protraction tactics until the argumentation game is over. In an example of using this strategy we show that it allows an agent to get a decision outcome that would be different without the use of the strategy.

Argumentation games

In this section, we present the basic terminology and definitions around argumentation games. Due to lack of space, we cannot provide complete formal definitions and concentrate only on the parts relevant for the following sections.

A *logic* $\mathcal{L}$ consists on the one hand side of a signature describing the allowed predicate and function symbols and variables and their arities and logical connectors and quantifiers to define what a well-formed formula is. In addition to this syntax, the semantics of $\mathcal{L}$ determine how truth values are assigned to formulae (usually using interpretations). Semantical consequence between a set of formulae Φ and a single formula α is expressed as $\Phi \models \alpha$ and it indicates that if all formulae in Φ are interpreted to true then also α will be interpreted to true.. Usually, $\models$ cannot be used for any automation, which means that a calculus and a consequence relation $\vdash$ based on this calculus is needed. And, for most logics and many applications, $\vdash$ needs to be resource bound to a $\vdash_{res}$ with res indicating the amount of "resource" available.

An agent $\mathcal{A}g$ consists of at least a knowledge base $KB_{\mathcal{A}g}$ consisting of formulae out of a logic $\mathcal{L}_{\mathcal{A}g}$, the ability to perform all of the actions required for the argumentation game (see below), which also reqires to apply a resource bound deduction $\vdash_{res}$, and the ability to determine what outcome of a given instance of the argumentation game it favors. The knowledge in the knowledge base $KB_{\mathcal{A}g}$ can be of different certainty (i.e. assumptions, beliefs and so on) and in order

to be able to deal with conflicts with other agents (which argumentation is all about) the agent needs to be willing to "accept" even knowledge contradicting its own knowledge from the outside that it cannot refute (for example in an argumentation game).

An argument $\langle \Phi, \alpha \rangle$ consists of a set Φ of formulae and a single formula α from a logic $\mathcal{L}$. We require that Φ is consistent (i.e. there is no contradiction between the formulae in Φ) and that $\Phi \vdash \alpha$ for a consequence relation for a calculus for $\mathcal{L}$ (but, naturally, agents willing to use malicious argumentation might ignore these requirements, so that checking them will have to be one of the actions an agent should do).

An argumentation game consists of two or more agents (in the following, we will only look at two) with some shared knowledge base KB_{shared}[1] that take turns at making an *utterance* (with a given maximum number of turns). An utterance consists of a performative identifying which type of utterance is being made, and possibly additional content such as arguments or references to other utterances, as defined by the performative being uttered. Two key performatives, that establish attacks on arguments in previous utterances, are *undercut* and *rebuttal*. If $a_1 = \langle \Phi, \alpha \rangle$ and $a_2 = \langle \Psi, \beta \rangle$, then a_2 is an undercut of a_1, if $\Phi \cup \{\beta\}$ leads to a contradiction and a_1 and a_2 are rebuttals of each other, if $\{\alpha, \beta\}$ leads to a contradiction. Another performative that will play a role in our strategies is the *redundant* performative that is used to identify that an opponent's argument was already used by the opponent before (which naturally requires that an agent is checking for this). Performing an argumentation game with particular agents, a particular KB_{shared} and a particular topic (which is usually established by the first utterance and the agents' intended outcomes) is also called a game instance.

While many argumentation games in literature do not set resource limits on the agents (except for limiting the number of turns), most practical decision making problems will have resource limits like a maximal time allowed for coming to a decision. There are several ways how limitations can be integrated into an argumentation game. We will in the following assume that each step of an agent in the game, i.e. determining an utterance and all parameters of it, has a resource limit. This is not a limitation for the following, because an agent naturally can use some of the resources for one step to already perform computations for future steps (although simple agents might not make use of this opportunity).

Malicious argumentation tactics

With limits on the resources an agent can use to evaluate an utterance of another agent and to determine its own next utterance, there are different ways to take advantage of the limitation. Making an argument that cannot be countered within the available resources is not the only way, it is also possible to create arguments that are only intended to squander resources and keep an opponent away from looking into "shaky" earlier arguments. In the following, we will present several tactics for creating arguments that produce various effects that can be combined over an instance of an argumentation game to win the game although objectively the game should be lost.

Superfluously complex argumentation tactics

The basic idea of these kinds of argumentation tactics is to rewrite an argument so that checking it for consistency or establishing that it indeed attacks another argument requires more resources than the opponent agent has available for doing so. As a result, depending on in what kind of utterance the argument is used, the argument is accepted (either to establish a fact or to counter another argument), despite it normally not being accepted if enough resources were available. Unfortunately, there are many possible ways to perform such a rewriting that creates superfluously complex arguments.

In general, it can be shown that any argument rewrite function f_{rew} that is targeted, scalable, validity preserving and semantic preserving can be used to create superflously complex arguments. Here, f_{rew} takes as arguments an argument $\langle \Phi, \alpha \rangle$, a target formula t and a resource bound res. In the following, let $\langle \Phi_{rew}, \alpha_{rew} \rangle = f_{rew}(\langle \Phi, \alpha \rangle, t, res)$. f_{rew} is targeted, if for $t \in \Phi$ we require $t \notin \Phi_{rew}$ and $\Phi_{rew} \vdash t$. f_{rew} is scalable, if it is targeted and for all values for res, we have that $\Phi_{rew} \nvdash_{res} t$. f_{rew} is validity preserving, if $\langle \Phi_{rew}, \alpha_{rew} \rangle$ is valid if and only if $\langle \Phi, \alpha \rangle$ is valid. And f_{rew} is semantic preserving, if $\Phi_{rew} \vdash \Phi$ and $\alpha_{rew} \vdash \alpha$. Please note that while it is naturally also possible to complicate α this would make it more difficult to test for it being an undercut or rebuttal, which naturally is not useful in an argumentation game.

In the following, we will present two rewrite functions for arguments that are generalizations of methods presented in (Kuipers and Denzinger 2010).

The *implication chaining* tactic makes use of the semantics of the $\rightarrow$ operator in most logics and the resources needed by the consequence relation to deal with certain formulae that have $\rightarrow$ as their main operator. The rewrite function $f_{rew,imch}$ for this tactic is defined as $f_{rew,imch}(\langle \Phi, \alpha \rangle, t, res) = \langle \Phi - \{t\} \cup \{\xi_0, \xi_0 \rightarrow \xi_1, ..., \xi_k \rightarrow t\}, \alpha \rangle$ where k is equal or greater than the minimal amount of deduction steps for $\rightarrow$ to use up res resources and the ξ_i are formulae fulfilling the following conditions: $t \neq \xi_0$, $t \neq (\xi_i \rightarrow \xi_{i+1})$ for all $i = 0, ..., k-1$, $\Phi \nvdash \xi_i$ and $\Phi \nvdash \neg\xi_i$ for all i, $\xi_0 \wedge ... \wedge \xi_k$ is not, with respect to the consequence relation, contradictory and $\xi_0 \wedge \ldots \wedge \xi_k \nvdash \alpha$. It can be shown that with all these conditions $f_{rew,imch}$ is targeted, scalable, validity preserving and semantic preserving. Even for these conditions there are still quite a lot of possibilities for choosing ξ_is. One of the easiest ways to create the ξ_is fulfilling the conditions is to simply make them "new" literals using predicates that are not in the signature of the other agent (which makes them (ontological) bullshit, see (Frankfurt 1986), and the end of the next subsection).

The *tautology injection* tactic makes use of formulae that are the semantical consequence of an empty set of formulae, so-called tautologies, and the usual semantics of the $\rightarrow$ and $\wedge$ operators to rewrite formulae into more complex, resource consuming formulae. The rewrite function $f_{rew,tain}$ for this tactic is defined by $f_{rew,tain}(\langle \Phi, \alpha \rangle, t, res) = \langle \Phi - \{t\} \cup$

[1]Note that while each agent's knowledge base has to contain KB_{shared} it does not have to be equal to KB_{shared}!

$\{(\tau_1 \wedge ... \wedge \tau_k) \rightarrow t\}, \alpha\rangle$, where $\vdash \tau_i$ for all i and k is the result of a so-called growth function such that deducing t from $(\tau_1 \wedge ... \wedge \tau_k) \rightarrow t$ using $\vdash$ requires more than res resources. Again, it can be shown that with these conditions $f_{rew,tain}$ is targeted, scalable, validity preserving and semantic preserving. Naturally, there are many tautologies for the usual logics and a malicious agent can easily have a list or even a list of tautology schemes (kind of patterns for formulae that guarantee that any instantiations of the scheme produce tautologies) to use for a particular rewrite.

Protraction argumentation tactics

In general, a protraction argumentation tactic is an utterance made by a malicious agent with the goal of wasting its opponent's resources by causing it to respond in a predictable manner. Usually, such a tactic is applied during an argumentation game when the malicious agent does not have a legitimate utterance to perform in the dialogue except for the "empty" utterance, which means that the agent was not successful in finding an (new) argument that attacks any of the opponent's arguments. Normally, without the protraction tactic, this would give the opponent the resources of its next move to look for counters of earlier utterances of the malicious agent, which would lessen the general problem of resource limits. There are many additional goals protraction tactics can fulfill, but the fulfillment of such additional goals depends on the reactions of the other agent, which can include to simply ignore the utterance created by the tactic. Therefore most such tactics construct utterances such that they still serve a purpose within the argumentation game, so that the other agent will have to address the constructed utterance.

The *syntactically redundant argument* tactic is the most simple protraction tactic and simply selects the argument from an earlier utterance of an agent and makes it the new utterance (using the same other parameters of the previous utterance). This tactic aims not so much at wasting resources for a step in the argumentation game but on wasting a step, since without countering this argument with the redundant performative (which is an utterance and therefore a step) this argument attacks the same opponent's argument as the argument that is copied.

The *semantically redundant argument* tactics have the same basic idea as the syntactically redundant argument tactic, namely selecting an argument from a previous utterance of the agent and kind of repeating it, but in contrast to the so easily detectable previous tactic, the semantically tactics rewrite the argument in such a way that the new argument is semantically equivalent to the one that is rewritten. This also guarantees the loss of a step in the game for the opponent, since the created argument needs to be countered, but if the opponent is not able to either counter this argument or realize that it is redundant the malicious agent might even get a second argument supporting its intended outcome. Even more, since deductions are needed by the opponent, the tactic also wastes resources. There is one potential drawback for this kind of tactic, namely that the malicious agent needs to create the semantically redundant argument, which has to involve some deductions on its own (and usually cannot be precomputed like a list of tautologies or tautology schemes). But the malicious agent still has the advantage (as so often is the case for "attackers") that it just needs to do some rewrite of the selected argument, while the opponent, in order to detect the semantical redundancy, has to check all possible rewrites.

There are many different ways how to create semantically equivalent formulae to a given formula. Perhaps the most simple such way is to use the fact that in most logics we have more operators than are really needed so that we can transform some of them into combinations of others while the resulting formula is semantically equivalent to the original formula. In propositional and first-order logic, for example, we can rewrite every formula (and every sub-formula of a formula) using double negation. Also, every formula of the form $A \rightarrow B$ is semantically equivalent to $\neg A \vee B$ and so on. And, naturally, these rewrites can be combined. We also can use the fact that $A \wedge true$ is equivalent to A to "expand" formulae using tautologies and then rewrite such expanded formulae to "break up" the tautology (for example: $A \rightarrow B$ can be extended with the tautology $C \vee \neg C$ to $(A \rightarrow B) \wedge (C \vee \neg C)$ which then can be rewritten into $(\neg A \wedge C) \vee (\neg A \wedge \neg C) \vee (B \wedge C) \vee (B \wedge \neg C)$). Naturally, it depends on the particular calculus used by the opponent agent how useful such rewrites are with regard to using up resources.

An *intentionally invalid argument* tactic aims at the one hand side to have the opponent agent waste resources in checking the presented argument for validity while on the other side, if the opponent agent is not doing this check or does not assign enough of its resources to it, attacking a previous argument of the opponent. There are several possibilities how such an argument can be constructed.

One possibility that does not require a lot of computation by the malicious agent is to use *random ontological bullshit*. Here "bullshit" is not used as an expletive, but in the meaning as a speech act first defined in (Frankfurt 1986). In general, a bullshit statement is a statement for which the entity uttering it neither beliefs that it is true nor beliefs that it is not true (in contrast to a lie which is a statement that the entity uttering it beliefs that it is false). As such, bullshit statements are an important source for malicious argumentation tactics, especially so-called ontological bullshit that is created by expanding the signature of the logic of the other agents by the malicious agent (which naturally means that the other agents do not know these symbols but assume that they are in the signature of the malicious agent and that the malicious agent includes everything necessary to "understand" these symbols in its arguments). If $\langle \Psi, \beta \rangle$ is the previous argument by the opponent that the tactic is supposed to target and $\gamma \in \Psi$, then we construct the argument $\langle \Phi, \alpha \rangle$ by setting $\alpha = \neg\gamma$ (which means we construct an undercut of the opponent's argument) and by setting Φ to a set of formulae that are constructed out of new symbols and that contain a formula δ and the formula $\neg\delta$. This means that Φ is contradictory, which makes the whole argument $\langle \Phi, \alpha \rangle$ invalid.

A malicious argumentation strategy

There are many possible ways to combine malicious argumentation tactics including also non-malicious arguments to create malicious argumentation strategies. Obviously, the knowledge about an opponent including what default decisions it makes when resources are exhausted, the opponent's actions and how weak the position of the malicious agent compared to the other agent is, influence what the best strategy for the malicious agent is to win the particular instance of an argumentation game. Given that malicious argumentation strategies is one of the contributions of this paper, we assume in the following that opponents will not employ their own malicious strategies and provide one example of a rather simple malicious strategy that we call *Exhaust and Protract*.

In general, any (malicious) argumentation strategy obviously has to play the argumentation game, which means it has to create a sequence of utterances. And each utterance in the sequence has to be created usually using all utterances in the game so far. As stated before, the creation of utterances can involve the use of (malicious) tactics and obviously there might be more than one tactic that is applicable at a particular point in the game. This means that the strategy not only has to determine all possible tactic instantiations that can be applied in the current situation (which usually also involves non-malicious ways of creating the next utterance) it also has to make the decision which of these instantiations really to use, which we call the control logic of the strategy.

As we have seen in the last section, malicious argumentation tactics come already with some conditions regarding when they can be applied. Usually the control logic of a strategy adds some more conditions just to tactic applications to filter out many, resp. favor some, of them to be able to consider only a few tactic instantiations for the final decision. Among these additional conditions can be that the opponent uttered an argument matching a particular pattern, that a specific turn in the game is reached or that the "normal" response utterance (without using any malicious tactics) would match a certain pattern or was not able to find an argument attacking any previous arguments of the opponent.

The general idea of Exhaust and Protract is to establish an argument in favor of the intended outcome of the argumentation game by the malicious agent (kind of gaining "the upper hand"), then using superflously complex argumentation tactics to attack any countering arguments by the opponent and if there is nothing to attack use protraction tactics to protect this intended outcome until the end of the argumentation game instance (essentially maintaining the upper hand).

More precisely, this malicious strategy is build around any basic non-malicious strategy with control logic *baseControl* that also uses a function *selectAttack* to select a former utterance of the opponent that it wants to attack with the next utterance. Additionally, the malicious strategy has a set T of formulae as argument that describes the possible values for the t parameter in the used argument rewrite functions for creating superfluously complex arguments (this is useful if these tactics require some pre-computing, because T then represents the formulae for which the resulting arguments just need to be looked up). Then the control flow for selecting the next utterance for the next turn by the Exhaust and Protract strategy is as follows:

First, determine what *baseControl* would do. If this is an attack of an opponents argument and if it contains an element t of T, use a superflously complex argumentation tactic with appropriate resource limit to rewrite this attack and use this new attack as result of the turn. If the result of *baseControl* does not include an element of T, use this as the result of the turn. If *baseControl* did not provide an attack utterance but did provide a non-empty utterance, then use this as result of the turn. If *baseControl* only comes up with the empty utterance and there is a possibility to apply a protraction tactic to a former attack utterance then apply this protraction tactic and the result is the result of the turn. Else use the empty utterance as result of the turn.

The Exhaust and Protract strategy as presented above makes a few assumptions about the opponent and the argumentation game that we would like to discuss further. First, in order for the exhaust part to work, we need the opponent to accept what an utterance with the superflously complex argument in it claims. This is known as the credulous assumption (Kuipers and Denzinger 2010). The opposite assumption is known as the skeptical assumption and it assumes that if resources are exhausted for any of the necessary tests of an argument and its utterance that the claims of the utterance are false (which naturally means that the argumentation game has to provide a performative for it and needs to also define consequences for uttering such a performative).

At first glance, it seems as if the skeptical assumption should be favored since it would prevent our Exhaust and Protract strategy to work. The problem is that making the skeptical assumptions can lead to "false positives", i.e. utterances with arguments that fulfill what the utterance claims but that, due to resource restrictions, would lead to a rejection of the utterance's claim. And, even worse, we can naturally use superflously complex argumentation tactics to create many such utterances, which essentially results in any agent who uses the skeptical assumption "leaving" many or all instances of argumentation games and therefore being "branded" unreasonable (when the game instances in which it rejected claims of utterances are analyzed with more resources and it is shown that the agent was wrong). So, with a perspective that is broader than performing a single game, there are strategies that can either drive away an agent using the skeptical assumption or force it to switch to the credulous assumption, which naturally then enables the Exhaust and Protract strategy.

Another assumption we made is that the malicious agent knows the resource limits of the opponent agent. Again, this is not a serious issue since we can extend the Exhaust and Protract with a starting phase that tries several resource bounds for the superflously complex argument tactic and switches over to the real exhaust part, when a bound was found (or uses the utterance that was not countered already as the "upper hand").

A use example

In the previous sections, we tried as much as possible to avoid committing to particular logics, particular agent features and particular argumentation games, which naturally can make it more difficult to understand the rather abstract concepts. In this section, we provide a use example of the Exhaust and Protract strategy for an application in arguing about department resources using first-order logic for the two involved agents.

More precisely, we will use the calculus consisting of Resolution and Factorization as consequence relation which allows us to determine if a set of formulae is inconsistent, respectively contradictory (by transforming all formulae into clauses and deducing the empty clause, see (Robinson 1965)). We have a manager agent, $\mathcal{M}an$ who is responsible for deciding which department in a company gets certain limited resources of the company (and shares the general philosophy of many accountants that no one should get anything from him, although, due to previous use of the skeptical assumption and consequent reprimands from the higher ups, he has to use the credulous assumptions). And we have an agent. $\mathcal{D}ep$, who represents one of the company's departments and wants access to a resource $Cres$. In fact, we will have two instances of $\mathcal{D}ep$, namely $\mathcal{D}ep_{norm}$ and $\mathcal{D}ep_{mal}$ who differ in the argumentation strategy used, with $\mathcal{D}ep_{norm}$ using a standard non-malicious strategy and $\mathcal{D}ep_{mal}$ using the malicious Exhaust and Protract strategy described in the last section. The knowledge base of both variants of $\mathcal{D}ep$ is $KB_{\mathcal{D}ep} = \{de_1\}$ with

$de_1 = Permission(\mathcal{M}an, Cres, \mathcal{D}ep)$

indicating that it believes that $\mathcal{M}an$ can give it (i.e. $\mathcal{D}ep$) permission to get resource $Cres$. The intended decision both variants of $\mathcal{D}ep$ would like as the result of the argumentation is

$de_2 = Allocate(Cres, \mathcal{D}ep)$

meaning that resource $Cres$ is allocated to $\mathcal{D}ep$.

$KB_{\mathcal{M}an}$ consists of three formulae:

$ma_1 = \forall p \forall r \forall g \neg Permission(p, r, g)$

indicating the already mentioned basic assumption of the manager that no one should be able to give anyone access to any resource,

$ma_2 = \forall p \neg Authority(p)$

meaning that (in the opinion of $\mathcal{M}an$) no one should have authority (see the common knowledge base below), and

$ma_3 = Restricted(Cres)$

stating that $Cres$ is a restricted resource.

The argumentation game played by $\mathcal{D}ep$ and $\mathcal{M}an$ has as common knowledge base $KB_{shared} = \{co_1, co_2\}$ with

$co_1 = \forall r \forall g(Restricted(r) \rightarrow (Allocate(r, g) \leftrightarrow \exists p Permission(p, r, g)))$

and

$co_2 = \forall p \forall r \forall g(Permission(p, r, g) \rightarrow Authority(p))\}$

indicating that for a restricted resource r it can be allocated to g if and only if there is a p that has permission to do so (first formula) and that anyone who has permission to assign any resource to anyone has authority.

For this simple game we have chosen to use as resource the number of deductions made during a turn. This is, in contrast to using run-time, always repeatable. We chose $res = 10$ and 10 is also the number of turns in the game, 5 for each of the agents.

If we play the argumentation game instance from above with the $\mathcal{D}ep_{norm}$ variant of the department representative we get the following sequence of utterances. In the first turn, $\mathcal{D}ep_{norm}$ utters the argument $\langle\{de_2\}, \{de_2\}\rangle$ which obviously is very self-serving. $\mathcal{M}an$ attacks this utterance with an attack utterance (which can be either an undercut or a rebuttal) with the argument $\langle\{co_1, ma_3, ma_1\}, \{\neg de_2\}\rangle$ which indeed can be proven to be valid and to attack the initial argument of $\mathcal{D}ep_{norm}$ (within the resource limit). $\mathcal{D}ep_{norm}$ then undercuts this argument with the argument $\langle\{\{de_1\}, \{\neg\neg de_2\}\rangle$ which is both valid and an attack, but which is then undercut by $\mathcal{M}an$ with the argument $\langle\{co_2, ma_2\}, \{\neg de_1\}\rangle$. In the next turn, $\mathcal{D}ep_{norm}$ undercuts the last utterance with the argument $\langle\{de_1, \{\neg(co_2 \wedge ma_2)\}\rangle$, which the manager can undercut by repeating the previous argument $\langle\{co_2, ma_2\}, \{\neg de_1\}\rangle$. At this point, $\mathcal{D}ep_{norm}$ has no arguments other than the empty argument left, which it utters and which leads to the utterance of the empty argument for the rest of the game by both parties. Since every argument of $\mathcal{D}ep_{norm}$, including its initial argument, was countered it lost the game and the decision is not to allocate the resource to it.

If we use $\mathcal{D}ep_{mal}$ as the variant of the department representative in the game instance, then the first two turns are the same as before. But in the third turn $\mathcal{D}ep_{mal}$ now undercuts the argument of the manager with a superfluously complex argument, namely one created using tautology injection. The tautology used is the most simple one there is, namely $T \vee \neg T$ and we use 3 instances of it, namely $T_1 \vee \neg T_1$, $T_2 \vee \neg T_2$ and $T_3 \vee \neg T_3$ using the newly introduced predicate symbols T_1, T_2 and T_3. The created argument is $\langle\{((T_1 \vee \neg T_1) \wedge (T_2 \vee \neg T_2) \wedge (T_3 \vee \neg T_3)) \rightarrow de_1\}, \{\neg ma_1\}\rangle$. This argument could be attacked by the argument $\langle\{co_2, ma_1\}, \{\neg(((T_1 \vee \neg T_1) \wedge (T_2 \vee \neg T_2) \wedge (T_3 \vee \neg T_3)) \rightarrow de_1\}\rangle$, but proving that this argument undercuts the malicious argument required in our Resolution theorem prover 23 deduction steps, way more than the limit of 10. As a consequence, $\mathcal{M}an$ can only give the empty utterance, which puts $\mathcal{D}ep_{mal}$ ahead in the game. This means that it is now time for the protraction part of the malicious strategy. Normally, an instantiation of the strategy will use the same protraction tactic, but, in order to provide an example for all the tactic types we presented before, we will use different tactics in the rest of the game instance.

In the fifth turn of the game, $\mathcal{D}ep_{mal}$ uses the syntactically redundant argument tactic with the additional goal to strengthen his attack on the manager's utterance in the second turn, which means it repeats the argument $\langle\{((T_1 \vee \neg T_1) \wedge (T_2 \vee \neg T_2) \wedge (T_3 \vee \neg T_3)) \rightarrow de_1\}, \{\neg ma_1\}\rangle$. $\mathcal{M}an$ can immediately counter this argument with an utterance claiming it is redundant (since it is exactly the same argument) or it can first use the resources for this turn to try to find an attack on the original argument from turn 3. But even the second alternative does not provide enough resources to find an attack, so that in both cases the utterance by $\mathcal{M}an$ will be the redundant performative and $\mathcal{D}ep_{mal}$ is

still "ahead in the game".

The next turn of $\mathcal{D}ep_{mal}$ uses a semantically redundant argument tactic that rewrites its successful argument from the third turn using several of the semantical equivalences known for first-order logic and uses it to, again, attack the utterance of the manager from the second turn. The rewritten argument is $\langle\{(\neg T_1 \wedge T_1) \vee ((\neg T_2 \vee \neg T_3) \wedge (\neg T_2 \vee \neg\neg T_3) \wedge (T_2 \vee \neg T_3) \wedge (\neg\neg T_2 \vee T_3)) \vee de_1\}, \{\neg ma_1\}\rangle$. If we consider each application of one of the equations describing semantical equivalence as a deduction step, then this brings us near the limit. But checking for redundancy has to search through all possible applications, which brings us over the resource limit, so that $\mathcal{M}an$'s next utterance will be the empty one, again, which also means that we now have two utterances attacking $\mathcal{M}an$'s attack of $\mathcal{D}ep_{mal}$'s goal.

For the final turn of $\mathcal{D}ep_{mal}$ in the game, we use an intentionally invalid argument tactic. Due to the earlier protraction tactics, there is still only one utterance by the manager agent that can be targeted. As described in the previous section, the first half of the argument will be a set of formulae that are created out of new symbbols (that we all called Bs or bs with an index) and we have chosen the following set: $\{Bs_1(bs_2), Bs_3(x) \vee Bs_4(x,y), \neg Bs_1(x)\}$. On first glance, it seems that this set does not fulfill the requirement of being contradictive, but since $\neg Bs_1(bs_2)$ is an instance of $\neg Bs_1(x)$ the requirement is fulfilled. The complete argument is $\langle\{Bs_1(bs_2), Bs_3(x) \vee Bs_4(x,y), \neg Bs_1(x)\}, \{\neg ma_3\}\rangle$.

For the final turn of the game, $\mathcal{M}an$ faces three utterances of $\mathcal{D}ep_{mal}$ attacking its attack of the initial argument of $\mathcal{D}ep_{mal}$. Regardless of how it uses its resources for this final turn, it can only attack one of those utterances, so that at the end of the game instance the initial argument of $\mathcal{D}ep_{mal}$, namely $Allocate(Cres, \mathcal{D}ep)$ has "survived" all challenges and therefore $\mathcal{M}an$ has to grant $\mathcal{D}ep$ access to $Cres$, which is naturally the oppositie outcome than the game with $\mathcal{D}ep_{norm}$.

Before we finish this section, we would like to comment on the fact that naturally not only the other agents have to act under resource limits but that also the malicious agent has to stay within these limits. If we look at the example applications of our malicious tactics above, then most of them can make use of pre-computed information, which naturally reduces resource consumption drastically. The tautology used in the third turn can be taken from a pre-computed list that also can already contain rewrites, which takes care of using the semantically redundant argument tactic. The contradictory set using new symbols is also something that can be taken from a pre-computed list. So, very little effort is needed during the game to implement this malicious argumentation strategy.

Related work

As already stated, most works in automated argumentation do not consider any resource limitations on agents for making an utterance and the favored logic used is propositional logic, which together naturally creates decidable argumentation games and eliminates the possibility for malicious argumentation. Even the introduction of bullshit into argumentation games was done using propositional logic, which results in less opportunities for creating such arguments, see (Caminada 2009). But, realistically, nearly every practical application of argumentation for decision making will need the expressiveness of at least first-order logic. This leaves (Kuipers and Denzinger 2010) as the only work we are aware of that looks into how to create malicious arguments that cannot be detected via resource limited deduction.

Interestingly, (Kontarinis and Toni 2015) addresses *detecting* malicious behavior by a party in a multi-party debate (which allows agents to make utterances whenever they want, without the concept of a turn), but the authors have only lies and hiding of information as malicious behavior and do not address resource limitations at all (in fact, their evaluation criteria are not looking into the arguments and the underlying logic at all, but instead at the number and frequency of utterances of a party). While this idea of profiling a party is definitely of interest in identifying malicious agents, obviously all possible ways for being malicious need to be integrated in a profile including what was presented in (Kuipers and Denzinger 2010) and in this paper.

Conclusion and future work

We introduced the concept of a malicious argumentation strategy that agents in automated multi-party decision making (under resource constraints) can use to have the decision going their way despite objectively the decision should have been different. Such strategies make use of different malicious argumentation tactics and we presented additional such tactics aiming at new effects on the argumentation as a whole. We also presented a concrete such malicious argumentation strategy, Exhaust and Protract, and an example how applying it can indeed change the outcome of decision from not being granted a resource to getting it.

Future work should be aimed to look into creating more malicious argumentation strategies to get a better picture of the breadth of possibilities these strategies can have. And, naturally, we want to look into ways how to detect and prevent the use of malicious argumentation strategies (like creating profiles out of the utterances made, as already mentioned, or using concepts like trust).

References

Caminada, M. 2009. Lies and bullshit; distinguishing classes of dishonesty, In Proc. Social Simulation WS at IJCAI.

Frankfurt, H. 1986. On bullshit, RARITAN-A QUARTERLY REVIEW. 6.2, 88–100.

Kuipers, A. and Denzinger, J. 2010. Pitfalls in Practical Open Multi Agent Argumentation Systems: Malicious Argumentation. In Proc. COMMA 2010, 323–334, IOS.

Kontarinis, D. and Toni, F. 2015. Identifying Malicious Behavior in Multi-party Bipolar Argumentation Debates, In Proc. EUMAS 2015, 267–278, Springer.

McBurney, P. and Parsons, S. 2002. Dialogue Games in Multi-Agent Systems. Informal Logic 22(3), 257–274.

Robinson, J.A. 1965. A machine-oriented logic based on the resolution principle. Journal of the ACM 12(1), 23–41.