Arguments and Artifacts for Dispute Resolution

Enrico Oliva
Mirko Viroli
Andrea Omicini
DEIS, ALMA MATER STUDIORUM—Università di Bologna
via Venezia 52, 47023 Cesena, Italy
E-mail:{enrico.oliva,mirko.viroli,andrea.omicini}@unibo.it

***Abstract*—In a social context cultural differences, individual interests, and partial awareness are often the causes of disputes. Alternative Dispute Resolution (ADR) is usually considered to be alternative to litigation, and can also be used to allow disputing parts to find an agreement. A dispute resolution is not an easy task and usually involves more entities including mediator or arbitrator with multiple dialogue sessions.**

In the paper we focus the attention on dispute resolution system in artificial society proposing a model and a technology to support the persuasive processes. The persuasion is the principal form dialogue used in an ADR system where agents exchange arguments to support their positions.

The general architecture proposed to build an ADR system exploits two artifacts abstractions – Co-Argumentation Artifact and Dialogue Artifact – that provide the right abstractions to coordinate the agents during the argumentative process. The technological support for the artifacts is provided by the TuCSoN infrastructure, also exploiting a meta-programming technique in Prolog. Finally, in the paper we present a simplified example of the execution of a persuasion dialogue ground on the commitments.

I. ALTERNATIVE DISPUTE RESOLUTION

People develop systems and methods in order to settle conflicts in a fair way. Human societies define norm systems, infrastructure (such as court) and methods (such as trial) to achieve the dispute resolution.

In a global business process scenario there is a increasing need of speed-up the processes, and to make faster the conflict resolution. The new systems have to support legal process when for instance a negotiation is broken, they have to combine mediation and legal service to avoid litigation.

Alternative Dispute Resolution (ADR) is usually considered to be alternative to litigation. It also can be used as a colloquialism for allowing a dispute to drop or as an alternative to violence. ADR is generally classified into at least four subtypes: negotiation, mediation, collaborative law, and arbitration. Walker and Daniels [1] underline that legal negotiation is a part of traditional dispute resolution system rather than a component of the ADR movement. The legal negotiation directly occurs among agents that represent the disputants in a context similar to a courtroom.

Arguments have a central role in the process of formalising legal system, and in the trial, too. The paper [2] contains a survey of logic in computational model on legal argument. The authors present the main architecture of legal arguments with a four layer architecture: 1) logical layer, 2) dialectical layer, 3) procedural layer, and 4) strategic layer. Disputants use arguments in order to persuade the other parts of the dispute and also the decision makers—juries, judges, clients and attorneys. In [3] the use of arguments in an ADR systems is considered, and an analysis of arguments in different contexts such as arbitration, mediation and multi-party facilitation is presented. Argumentation plays an important role in conflict resolution systems, where it drives the ADR to obtain a successful solution of the dispute. The argumentation process promotes the values of justice, equality and community that are desirable in a dispute resolution system.

In an open agent society, the same issue as in human society holds: it is undesirable to resolve dispute by litigation. The development of a system for internal resolution of disputes in virtual organisations is presented by Jeremy Pitt et al in [3], which proposes a norm-government MAS and an ADR protocol specification for virtual organization exploited by intelligent agents.

ADR supplies a theoretical bases for Online Dispute Resolution (ODR) as defined in [4]. ODR has the purpose to extend the ADR process, moving it towards virtual environments while providing computation and communication support. In ODR, the role of technology used to facilitate the resolution of disputes between parties is crucial. It provides a structured communication, as well as an informed environment that helps to the successful conclusion of the conflict.

ODR could be seen as an instance of an ADR system, with a communication infrastructure and Artificial Intelligence (AI) techniques aiming at supporting the parties toward agreements. The reasoning and argumentation capabilities of the parties are achieved by exploiting AI methods.

Walton and Godden [5] show that argument-based dialogue, in particular persuasion dialogue, contributes to the construction of effective dispute resolution system. The main type of dialogue usually considered by ADR is negotiation, which could be interpreted as a particular sort of communication for the purpose of persuasion. In argumentation theory both types of dialogues are present: persuasion dialogue and negotiation dialogue. These two types of dialogue have a different structure and different goals, and in the context of ODR systems should be managed by different procedural rules.

A fundamental problem in ODR and ADR systems is that it is difficult to structure and process the information exchanged between negotiating parties. In order to resolve this problem

in this work we propose to build a ADR system based on the A&A meta-model [6] with Co-Argumentation Artifact (CAA) [7] and Dialogue Artifact (DA) [8] abstractions. We aim at providing a framework for conflict resolution in an agent-based society supplying a supporting infrastructure in order to manage arguments, to retrieve information and to bargain.

Our framework provides structured information based on logic tuple along with the control of dialogue processes through a mediated form of communication over a programmable infrastructure. These two features are useful in order to build MAS in a scalable and flexible architecture, and also to build ADR that supports multi-party dialogue sessions.

The aim of the work is to provide a more formal (functional) connection among the two types of argumentation artifacts CAA and DA in order to support a dialogue for dispute resolution. In particular we make explicit a set of functionalities useful during the dialogue to control the relation with the argumentative commitment store. For the CAA we collect a list of operations to manage a commitment store based on the argumentation system. On the other hand for the DA we describe by operational semantics the use of the CAA operations during the dialogue.

The result is a powerful architecture where it is possible to specify a dialogue grounded on the state of the commitment store enabling a partially automate dialogue execution through DA and CAA infrastructure. Using that dialogue specification the DA can automatically drive the sequence of actions based on the state of the CAA.

In Section II we introduce the architecture of the framework with the definition of the specific CAA and DA; in Section III we explain the argumentation and dialogue system by introducing the new operators to describe the interaction with the commitment store; finally in Section IV we present the case study, implementing a persuasion dialogue protocol.

II. Architecture

We propose our architecture for MAS based on A&A meta model to design a ADR/ODR application. An ADR system, especially on-line, exploits the forms of negotiation, arbitration or mediation required to achieve a solution. There, typically, the entities involved are more than two: at least, two participants and a third entity to help the dispute resolution such as in mediator and arbitrator procedure. The parties involved choose the procedure, terms and conditions of their dispute. For instance, in [3] an arbitration protocol is presented, along with concepts for decision making through formation and voting protocol.

In order to find a solution, the parties have the possibility to share any pertinent argument, make demands and evaluate the acceptability of an argument with respect to normative context. To do that, a multi-party dialogue protocol is required, and also an impartial computation over the shared knowledge. When the dispute involves an increasing number of participants it is necessary to introduce a mediated form of communication in order to have a scalable system. The essential point, here, is that in the act of mediating there are a number of evaluations that could be done automatically.

In that scenario our architecture provides the required abstractions: *(i)* Dialogue Artifact, *(ii)* Co-Argumentation Artifact to made a flexible system. In the DA we store the arbitration, mediation or negotiation protocol. The parties exploit the DA to take part of the discussion, which drives the dialogue ground on the commitments. The advantages are: the management of dialogue between multiple entities, and the automatic interaction with commitment/argument store. The CAA provides the right abstraction to made a commitment/argument store where it is possible evaluate automatically the argument validity respect of normative context. Also, it provides default function to exchange information, data and arguments, and to record their public commitments in private or public form. For instance, in a bargain among three or more entities handled by a CAA, the final set of arguments stored in the CAA during the bargain represent a form of contract among the parties.

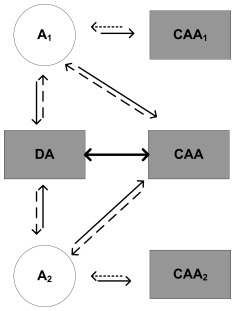

Fig. 1. General architecture of multi-agent argumentation system

A possible architecture for a Multi-agent argumentation system is shown in Figure 1 where A_1 and A_2 represent two rational agents. The suggested architecture exploits both local and global DA and CAA. The global CAA and DA provide services and functionalities for the entire agent society. Ideally, in the model, DA and CAA are separate entities with different and orthogonal functionalities. However, in an actual implementation, both shared artifacts could collapse in one unique global entity without loss of generality. The local CAA_1 and CAA_2 are used by agents in order to coordinate their mental state. Classically, those functions are provided by an internal argumentation component hidden inside the agent.

In the following, we focus our attention on the persuasion dialogue that is among the most common and useful dialogue in ADR. An interesting observation in [5] put in evidence the fact that a negotiation dialogue could naturally include or shift to persuasion dialogue in two points: 1) to follow an offer, and 2) to follow a rejection of an offer. In both cases reasons (by argument) are provided to prove the (un)acceptability of an offer. A dialogue model for persuasion could be composed of: 1) a commitment store for each participant, 2) an inference rule to draw conclusion from commitments in the commitment store made by the participant, and 3) practical rules that govern the sequence of locutions and their consequence. We

foresee that our architecture provides the desired abstraction and properties to implement the persuasion dialogue with agreement purpose in an agent society.

A. Co-Argumentation Artifact

The CAA provides co-ordination services to agents, allowing them to share, store and exchange arguments with one another working as a commitment store. In particular, for persuasion dialogue we exploit the ability of the CAA to automatically calculate argument and belief acceptability according to the agent attitudes and the argumentation semantics. In [9] are introduced agent attitudes in order to provide some acceptability criteria. An agent may have one of three acceptance attitudes about proposition: *(i)* a *credulous* agent can accept any formula for which there is an argument S; *(ii)* a *cautious* agent can accept any proposition for which there is an argument if no stronger rebutting argument exists; *(iii)* a *skeptical* agent can accept any proposition for which there is an acceptable argument S.

Exploiting our argument definition 1 and referring to our argumentation system in section III-A, we resolve the argument acceptance problem following the preferred semantics on argumentation: an argument is credulous acceptable if it belongs to some preferred extension; an argument is skeptical acceptable if it belongs to all preferred extensions.

The CAA validates the argument committed verifying their correctness and also it evaluates their acceptability verifying which of preferred sets belong to. The state of a CAA is represented by a collection of arguments with also the related list of conflict free sets, admissible sets and preferred extension. These sets are update for each argument insertion or removal.

In tuple notation these sets are expressed by a tuple name like `conflictfree`, `admissible` and `preferred` and a parameter composed by a list of lists of argument names indicated by `arg1,...,argN` i.e. `conflictfree([[arg1,...,argN],...])`. In particular for persuasion dialogue the CAA supports the agent `credulous` and `skeptical` attitudes, calculating in which argument set the reference argument belong to. Here, we list a set of operation provided by the CAA where Arg parameter means a generic input argument:

- $acceptable(Arg, Attitude)$: the CAA verifies the acceptance of the argument Arg respect of state of the commitment store with the type of acceptability specified by $Attitude$
- $read(ArgTemplate)$: the CAA returns an argument that logically unifies with $ArgTemplate$
- $conflict(Arg)$: the CAA verifies the existence of an argument $\in$ CAA in rebuttal relation with Arg (see section III-A)
- $attack(Arg)$:the CAA verifies that Arg is in undercut relation with an argument $\in$ CAA (see section III-A)
- $defeat(Arg)$ the CAA verifies the existence of an argument $\in$ CAA in undercut relation with Arg
- $remove(Arg)$: the CAA removes the argument Arg
- $commit(Arg)$: the CAA stores the argument Arg and it recompute the conflict free sets, the admissible sets and the preferred extensions

For further details, including an implementation and examples of this argumentation framework, we refer to the paper [10].

B. Dialogue Artifact

The DA is the abstraction to encapsulate the rules of dialogue and it coordinates the entities during persuasion process. We follow the definition provided of this artifact in [8] where the DA is composed of three components: a collection of specifications of dialogue protocols; a collection of commitments stores; and a collection of specifications of interaction control. Basically, the function of the DA is to drive the agents general type of dialogue keeping trace in the commitments stores of the partial results of the communication. Moreover the DA keep in charge to suggest of the agents the possible right moves constrained by the state of the commitment store. Here a list of operation provided by DA:

- *nextlocutions([L])*: the DA provides the list of possible locutions
- *lastlocution(L)*: the DA provides the last locutions
- *state(S)*: the DA provides the protocol state
- *act(L)*: the DA store the locution L and updates the state of protocol
- *cs(A)*: the DA executes an action A over the commitment store

From a general architecture point of view the commitment store of DA is provided by CAA correctly implemented and the global state of the system is represented by the state of the CAA and the state of the protocol in the DA.

III. Argumentation & Dialogue System

In order to achieve agreement among agents a common dialogue system and a shared argumentation system are required in the agent society. Following, we propose a formalization of both systems.

A. Argumentation System

Our reference argumentation system is introduced in detail in [7] as an extension of the Dung's framework [11] with the definition of the structure inside the arguments. Here we report briefly the argument definition and the object language.

The object language of our system is a first-order language, where Σ contains all well-formed formulae. The symbol $\vdash$ denotes classical inference (different styles will be used like deduction, induction and abduction) $\equiv$ denotes logical equivalence, and $\neg$ or non is used for logical negation.

Definition 1 (argument): An *argument* is a triple $A = \langle B, I, C\rangle$ where $B = \{p_1, \ldots, p_n\} \subseteq \Sigma$ is a set of *beliefs*, $I \in \{\vdash_d, \vdash_i, \vdash_a\}$ is the inference style (respectively, *deduction*, *induction* or *abduction*), and $C = \{c_1, \ldots, c_n\} \subseteq \Sigma$ is a set of *conclusions*, such that:

1) B is consistent
2) $B \vdash_I C$

Deductive Inference		Inductive Inference	
MP	$\frac{A \quad A \to B}{B}$	θ-su	$\frac{B}{R} \quad where \quad R\theta \subseteq B$
MT	$\frac{\neg A \quad B \to A}{\neg B}$	Abductive Inference	
MMP	$\frac{B_1,\ldots B_n \quad (B_1,\ldots B_n) \to C}{C}$	Ab	$\frac{B \quad A \to B}{A}$

TABLE I
DEDUCTIVE INFERENCE: (MP) MODUS PONENS, (MMP) MULTI-MODUS PONENS AND (MT) MODUS TOLLENS. INDUCTIVE AND ABDUCTIVE INFERENCE: (θ-SU) θ-SUBSUMPTION, (AB) ABDUCTIVE

3) B is minimal, so no subset of B satisfying both 1 and 2 exists

The types of inference I we consider for deduction, induction and abduction are shown in Table I. Modus Ponens (MP) is a particular case of Multi-Modus Ponens (MMP) with only one premise. The inference process θ-subsumption derives a general rule R from specific beliefs B, but is not a legal inference in the strict sense.

Definition 2 (contrary): The contrary (or attack) relation R is a binary relation over Σ that $\forall p_1, p_2 \in \Sigma$, $p_1 R p_2$ iff $p_1 \equiv \neg p_2$.

For defeat of arguments there are two possible types of attack based on the contrary relation: 'conclusions against conclusions', called *rebuttals*, and 'conclusions against beliefs', called *undercuts*.

Definition 3 (undercut): Let $A_1 = \langle B_1, I_1, C_1 \rangle$ and $A_2 = \langle B_2, I_2, C_2 \rangle$ be two distinct arguments, A_1 is an *undercut* for A_2 iff $\exists h \in C_1$ such that hRb_i where $b_i \in B_2$.

Definition 4 (rebuttal): Let $A_1 = \langle B_1, I_1, C_1 \rangle$ and $A_2 = \langle B_2, I_2, C_2 \rangle$ be two distinct arguments, A_1 is a *rebuttal* for A_2 iff $\exists h \in C_1$ such that hRc_i where $c_i \in C_2$.

The definitions of conflict-free set, admissible set, preferred extension are the basic ones in our argumentation system. These sets are composed of arguments that together feature different kinds of properties like absence of conflicts or common defence, formally introduced in [11].

We consider also important argument extensions such as acceptability in order to determine whether a new argument is acceptable or not. An argument is acceptable in the context of preferred semantics if an argument is in some/all preferred extensions (credulous/skeptical acceptance).

B. Dialogue System

Our intention here is to capture the rules that govern legal utterance and the effect of the utterances on the commitment store of the dialogue. We use a process algebra approach to represent the possible paths that a dialogue may take, and to represent explicitly the operations to and from the commitment store. Our *communication language* is a set of locutions L_c where a locution is a expression of the form $perf_{name}(Arg_1, \ldots, Arg_n)$ in particular $perf_{name}$ is a performative and Arg_x is either a fact or an argument. An agent performing a dialogue using the communication language can utter a locution composed of facts and arguments. An argument is represented with the tuple `argument(B,I,C)`; also a fact is considered an argument but with an (true) implicit premise and it is represented by syntax `argument(true,I,C)`.

Definition 5 (action): An *action* A is defined by the syntax $A ::= s : L_c | s[t_1, \ldots, t_n] : L_c$ where s indicates the source, and $[t1, \ldots, t_n]$ indicates the (optional) targets of the message. On the other, beyond this, we include additional atomic operations K over commitment stores—many of them can actually occur into one argumentation artifact.

Definition 6 (term action): A *term action* K has the syntax $K ::= commit(C, X) | read(C, X) | conflict(C, X) | attack(C, X) | defeat(C, X) | acceptableS(C, X) | acceptable(C, X)$, where C is a term representing the commitment store identifier, and X is a term representing the commitment.

A protocol P specifies by standard process algebra operator $(., +, \|)$ respectively sequence, parallel and choice, the set of actions and term actions that the agents and DA might execute. For example, an abstract dialogue protocol definition is given by $D := s : a_1 \parallel s : a_1 \parallel s : a_1 \parallel t : a_2 \parallel t : a_3$ where agent s invokes a_1 three times, agent t can invoke a_2 and a_3 only once, but in whichever order. For more detailed protocol definition with the process algebra approach and the related operational semantic we refer to the work [8].

Enriching the previous work we augment the set of K term actions in order to clarify the relations with a commitment store based on arguments. The behaviour of term actions is defined by operational semantics. This semantics describes the evolution over time of the dialogue state and the states of commitment store (seen as the composition of all commitment stores). In essence, the commitment store is the knowledge repository of the dialogue as a whole, and it is expressed in our framework as a multiset of arguments.

Definition 7 (commitment store): A commitment store C is a multiset of arguments and it is defined by the syntax $C ::= 0 | (C|C) | X$ where X is a argument, and 0 is the empty set.

We use also notation $t\{x/y\}$, to mean term t after applying the most general substitution between terms x and y—x should be an instance of y, otherwise the substitution notation would not make sense. Finally, we define the semantics of K operation that describe the interaction and evolution over time of the commitment store C in function of protocol P:

$$(C)commit(x).P \xrightarrow{\tau} (C'|x)P \quad (1)$$

$$(C|x)read(y).P \xrightarrow{\tau} (C|x)P\{x/y\} \quad (2)$$

$$(C|x)remove(y).P \xrightarrow{\tau} (C)P\{x/y\} \quad (3)$$

$$(C|x)conflict(y)).P \xrightarrow{\tau} (C|x)P \text{ if } \{x \text{ rebuttal } y\} \quad (4)$$

$$(C|x)attack(y)).P \xrightarrow{\tau} (C|x)P \text{ if } \{y \text{ undercut } x\} \quad (5)$$

$$(C|x)defeat(y)).P \xrightarrow{\tau} (C|x)P \text{ if } \{x \text{ undercut } y\} \quad (6)$$

$$(C|\mathcal{E})acceptS(y).P \xrightarrow{\tau} (C|\mathcal{E})P \text{ if } \{\forall E \in \mathcal{E}, y \in E\} \quad (7)$$

$$(C|E)acceptable(y).P \xrightarrow{\tau} (C|E)P \text{ if } \{y \in E\} \quad (8)$$

As usual, we write $s \to is'$ in place of $\langle s, i, s' \rangle \in \to$, meaning the dialogue system moves from state s to s' due to interaction i—either an action a, or an internal step τ (an operation over the commitment store).

Rule (1) provides the semantic of `commit` operation, expressing that x term is added to the commitment store C,

and the state of commitment store is updated recalculating conflict free set and preferred extensions after that the process continuation can carry on. Rules (2) and (3) to read and remove terms from commitment store C: the use of substitution operator guarantees that the term x in the commitment store is an instance of the term x to be retrieved. Rules (4), (5) and (6) provide the semantics for attack, conflict and defeat relations using the standard definition of undercut and rebuttal. Finally, rules (7) and (8) express the semantics for acceptable operators for skeptical(7) and credulons(8) acceptance, where $\mathcal{E}$ is the set of all preferred extension E in the commitment store C.

IV. Persuasion Dialogue Application

In persuasion dialogue the goal of a participant is to prove his/her thesis and to rationally persuade the other parties. With the word "persuasion" we mean not a psychological persuasion but rather a rational persuasion supported by arguments. Walton & Krabbe [12] observe that disputes is a subtype of persuasion dialog—where the parties disagree about a single proposition φ. So, for instance, at the beginning of the dialogue a party beliefs in φ while the other belief in $\neg\varphi$, so they have a contrary opinion about a proposition. Generally the following moves are allowed in the dialogue: asking question, answering question, and putting forward arguments. Following Walton [5], a proponent in a persuasion dialogue has successful when: 1) the responded has committed all the premises of the argument 2) each argument is corrected 3) the chain of argument has the proponent thesis as its conclusion

In [13] is presented a survey of formal system of persuasion dialogue that point out the crucial role of the regulating interaction among agents rather than design of behaviour in individual agent within a dialogue. Among the main approaches to design persuasion dialogue and communication between agents based on arguments we draw inspiration from the Parson and McBurney's [9] and Prakken's [14] approaches; and also from [15], where the authors show how each move of a dialogue could be specified by rationality rules, dialogue rules and update rules explicating the relation with the commitment store.

The more common locutions of persuasion dialogue that can be found in literature are well collected in [13], and briefly listed here:

- *claim* φ (assert): The agent asserts a formula φ to start the persuasion.
- *why* φ (challenge): The agent asks for reasons about the φ formula.
- *concede* φ (accept): The agent accepts the validity of φ.
- *reject* φ(retract): The agent no commits the φ. In some cases it retracts the formula from the commitment store previously stored.
- S *since* φ (argue): The agent provides reasons for φ formula by an argument.

```
dialog_persuasion(X,Y,P):=
    X:assert(argument(true,I,P)).
    dialog_response(X,Y,argument(true,I,P))

dialog_response(X,Y,argument(true,I,P)):=
    Y:accept(argument(true,I,P)) +
    Y:reject(argument(true,I,P)) +
    Y:why(argument(true,I,P)).
       X:argue(argument(B,I1,P)).
       dialog_argue(X,Y,argue(argument(B,I1,P))).

% Evaluation of chain argument support of P assertion
dialog_argue(X,Y,argument(B,I,P)):=
    Y:accept(argument(B,I,P)) +
    Y:reject(argument(B,I,P)) +
    Y:argue(argument(B1,I1,P1).(
       X:retract(P) +
       X:argue(argument(B2,I2,P2)).
       dialog_argue(X,Y,argument(B,I,P)))).
```

Fig. 2. Persuasion dialogue without interaction with the CS

A. Protocol Specification

In order to make a persuasion dialogue concrete, a persuasion protocol is typically to be defined among two parties—proponent and respondent. We formalise through our process algebra a generic persuasion dialogue with and without automatic action to the commitment store. The protocol draws inspiration from [16, 15] and adds repetition rule proposed by [13]. The dialogue could be partially driven through the state of commitment store by the actions listed in II-A that are specifiable in the protocol.

Figure 2 shows a dialogue protocol for persuasion where an agent can accept or reject an assertion P based on its attitudes by an internal evaluation of facts and argument acceptability. Then an argumentation phase starts that concludes with either an acceptance or rejection of the assertion P expressed by an argument with "true beliefs". The relation among dialogue and commitments is not explicitly expressed. In a dialogue, each move could be specified by rationality rules, dialogue rules and update rules [15]: the rationality rules specify the pre and post conditions for playing a move; The update rules specify the modification of commitment store; And the dialogue rules specify the next moves. With our process algebra we have the expressive power to cover the three types of dialogue rules. For instance, we propose modified version of the persuasion protocol in the figure 3 where we provide the specification of the automatic evaluation of some preconditions (rationality) and the consequent modification of the commitment store (update). In that version of the dialogue specification the DA automatically drives the sequence of action through the state of the commitment store using the term actions: *commit* and *acceptable*. In the choice points some locutions are automatically chosen by preconditions based on the state of acceptability of arguments. In particular the proponent agent (X) is constrained to retract the proposal if its supporting argument is not acceptable during the arguing phases. Also, the opposer (Y) is constrained to accept the proposal if its opposing argument is not acceptable respect to the state of the commitment store. We exploit the ability of the CAA in order to find argument acceptability following the credulous

argumentation semantic.

This protocol formalisation is very flexible, and opens a number of different courses of actions. The problems could be the termination of dialogue and the determination of the dialogue result. The dialogue is partially automated through DA and CAA infrastructure. Agents have the time the control over own actions, and can decide in every moment to suspend the dialogue.

```
dialog_persuasion(X,Y,P):=
  X:assert(argument(true,I,P)).
  dialog_response(X,Y,argument(true,I,P))

dialog_response(X,Y,argument(true,I,P)):=
  Y:accept(argument(true,I,P)).commit(argument(true,I,P)) +
  Y:reject(argument(true,I,P)) +
  Y:why(argument(true,I,P)).
     X:argue(argument(B,I1,P)).commit(argument(B,I1,P)).
     dialog_argue(X,Y,argue(argument(B,I1,P)))

% Evaluation of chain argument support of P assertion
dialog_argue(X,Y,argument(B,I,P)):=
  Y:accept(argument(B,I,P)).commit(argument(B,I,P)) +
  Y:reject(argument(B,I,P)) +
  Y:argue(argument(B1,I1,P1)).commit(argument(B1,I1,P1)).(
     acceptable(argument(B1,I1,P1)).(
       X:retract(argument(B,I,P)) +
       X:argue(argument(B2,I2,P2)).commit(argument(B2,I2,P2)).(
         acceptable(argument(B2,I2,P2)).
         dialog_argue(X,Y,argument(B,I,P)) +
         not(acceptable(argument(B2,I2,P2)).
         X:retract(argument(B,I,P))
       )
     ) +
     not(acceptable(argument(B1,I1,P1))).
       Y:accept(argument(B,I,P)).commit(argument(B,I,P))
  )
```

Fig. 3. Persuasion dialogue with CS interaction: Automatic evaluation of acceptability

B. Technology Support

Logic programming and meta logic programming are two useful techniques to prototype quickly complicated software systems with rational behavior. The technological support to build artifacts is provided here by TuCSoN, a coordination infrastructure for MAS introduced in [17]. TuCSoN infrastructure following a Linda like coordination model provides a programmable environment based on logic tuples.

To realize CAA and DA implementing the necessary operators listed in section II, an obvious choice is to exploit a TuCSoN logic tuple centre. In fact, on the one hand a typical argumentation process is composed of two parts: (1) knowledge representation; and (2) computation over the set of arguments. On the other hand, the tuple centre architecture is also composed of two parts: an ordinary tuple space where the information are stored in form of tuples, and a behaviour specification that defines the computation over the tuple set. Thus, a TuCSoN tuple centre could support the argumentation process by representing knowledge declaratively in terms of logic-tuple arguments, and by specifying the computation over argument set in term of ReSpecT specification tuples.

From a practical point of view, we exploit the Prolog language and ReSpecT to implement the meta programs for managing the argument set with the ability to calculate: (1) the argument validity; (2) the relations of undercut and attack between argument; (3) the conflict-free sets; and (4) the preferred extensions. Each argument has its own context, where the argument is true. The context is provided in the argument and is composed only by the set of beliefs – facts and rules – directly declared in the tuple. The connection between the premises and the conclusion is expressed in terms of the corresponding inference process, which is specified in the argument too.

The meta programs are useful also to realize the control of dialogue interaction. The engine of process algebra management is implemented exploiting a transition system defined in Prolog by the predicates `transition(Currentstate, Action, Newstate)`. The program has to have the ability to change dialogue state after an agent action, to search of next admissible move after an agent request, and also to make the automatic interaction with the commitment store by argumentative actions. For a more detailed explanation of the use of meta-program technique to manage argument and dialogue process we forward the interested reader to [18].

C. Example of a Run

In this section we provide an example of run of a simplified version of persuasion dialogue exploiting the TuCSoN infrastructure and showing its use. In order to perform the dialogue simulation TuCSoN provides useful tools: *CLIAgent* to simulate agents interaction and *Inspector* to inspect current state of tuple space . The Inspector tool shown in figure 4 allows users to observe and debug the communication state and the behaviour of a tuple centre. In particular, it makes possible to inspect the tuple set, the pending query set, the triggered reaction set, and the behaviour specification set.

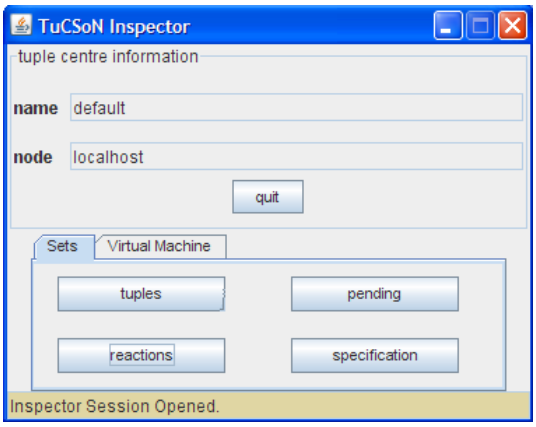

Fig. 4. Inspector tool

The *CLIAgent* tool allows users to invoke the commands of the TuCSoN coordination language. For our purpose we exploit the *CLIAgent* to utter agent locution in the form `out(move(Dialog,Id,Locution))`.

The rules to manage the dialogue in the DA are programmed with the ReSpecT code in [18]; for the commitment store

the same tuple space of dialogue is considered, and the initial dialogue state is expressed by the tuple

```
dialogstate(persuasion,[act(X,assert1(P)),
  (act(Y,accept(P))+act(Y,reject(P)))+act(Y,assert1(non(P)))+
  act(Y,why(P),act(X,argue(argument(N,bel(B),inf(I),conc(C)))),
  (act(Y,accept(N))+ act(Y,reject(N)))]).
```

The locutions that could be uttered in that dialogue are: *assert*, *accept*, *reject*, *why*, and *argue*. We start the simulation sending a *assert* locution in tuple centre from agent *Paul* by the CLIAgent shown in the figure 5. After that move, the infrastructure reacts and calculates next dialogue state.

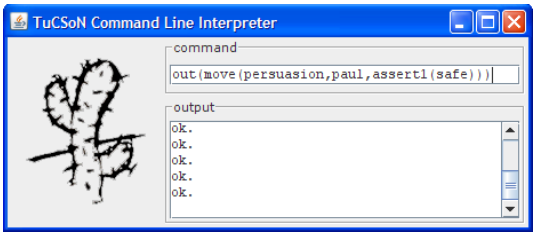

Fig. 5. CLIAgent

```
move(persuasion,paul,assert1(safe))
dialogstate(persuasion,
  ['+'('+'('+'(act(_4,accept(safe)),act(_4,reject(safe))),
  act(_4,assert1(non(safe)))),act(_4,why(safe))),
  act(paul,argue(argument(_3,bel(_2),inf(_1),conc(_0)))),
  '+'(act(_4,accept(_3)),act(_4,reject(_3)))])
```

The responder agent *Olga* can ask the possible admissible next locutions by the tuple `rd(nextlocutions(persuasion,L))`, and the tuple centre responds by new tuple `nextlocution`.

```
nextlocution(persuasion,
  [act(_2,accept(safe)),act(_2,reject(safe)),
  act(_1,assert1(non(safe))),act(_0,why(safe))])
```

At this point, the responder chooses a move either from the state of commitment store or independently from our knowledge base—for instance in this case the choice could be *why(safe)*. Figure 6 shows the state of the tuple centre after *Olga* locution by the inspector tool. The new `dialogstate` expresses the remaining locution constrained by previous logical unification of *paul* and *olga* identifiers.

```
dialogstate(persuasion,[act(paul,
  argue(argument(_3,bel(_2),inf(_1),conc(_0)))),
  '+'(act(olga,accept(_3)),act(olga,reject(_3)))])
```

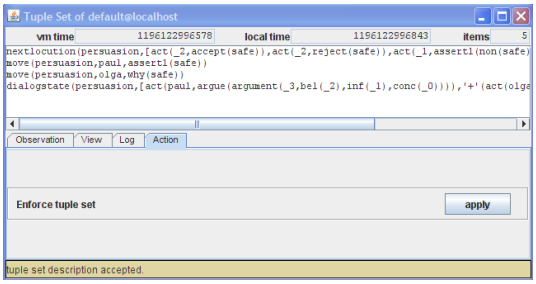

Fig. 6. Tuple Set

References

[1] G. B. Walker and S. E. Daniels, "Argument and alternative dispute resolution systems," *Argumentation*, vol. 9, no. 5, pp. 693 – 704, 1995. [Online]. Available: http://www.springerlink.com/content/m1263hp73g344127

[2] H. Prakken and G. Sartor, *Computational Logic: Logic Programming and Beyond. Essays In Honour of Robert A. Kowalski, Part II.*, 2048th ed., ser. Lecture Notes in Computer Science 2048. Berlin: Springer, 2002, ch. The Role of Logic in Computational Models of Legal Argument: A Critical Survey, pp. 342–380. [Online]. Available: http://www.springerlink.com/content/e0j2bhdq8gm8cg98

[3] J. Pitt, D. Ramirez-Cano, L. Kamara, and B. Neville, "Alternative Dispute Resolution in Virtual Organizations," in *Proceedings of The Eighth Annual International Workshop "Engineering Societies in the Agents World" (ESAW 07), Athens, Greece*, 2007.

[4] T. Schultz, G. Kaufmann-Koheler, D. Langer, V. Bonnet, and J. Harms, "Online dispute resolution: State of the art, issues, and perspectives," Faculty of Law and Centre Universitaire Informatique, University of Geneva, Tech. Rep., October 2001, draft Report.

[5] D. Walton and D. M. Godden, "Persuasion dialogue in online dispute resolution," *Artificial Intelligence and Law*, vol. 13, pp. 273–295, 2005. [Online]. Available: http://www.springerlink.com/content/k813173822154532

[6] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the A&A meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, Dec. 2008, special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

[7] E. Oliva, P. McBurney, and A. Omicini, "Co-argumentation artifact for agent societies," in *Argumentation in Multi-Agent Systems*, ser. LNAI, S. Parsons, I. Rahwan, and C. Reed, Eds. Springer, Apr. 2008, vol. 4946, ch. 3, pp. 31–46, 4th International Workshop (ArgMAS 2007), Honolulu, HI, USA, 15 May 2007. Revised Selected and Invited Papers. [Online]. Available: http://www.springerlink.com/content/5817w1n882861170/

[8] E. Oliva, M. Viroli, A. Omicini, and P. McBurney, "Argumentation and artifact for dialogue support," in *5th International Workshop "Argumentation in Multi-Agent Systems" (ArgMAS 2008)*, I. Rahwan and P. Moraitis, Eds., AAMAS 2008, Estoril, Portugal, 12 May 2008, pp. 24–39.

[9] S. Parsons and P. McBurney, "Argumentation-based communication between agents," in *Communication in Multiagent Systems*, ser. LNCS, M.-P. Huget, Ed., vol. 2650. Springer, Berlin, September 2003, pp. 164–178.

[10] E. Oliva, P. McBurney, and A. Omicini, "Co-

argumentation artifact for agent societies," in *4th International Workshop "Argumentation in Multi-Agent Systems" (ArgMAS 2007)*, S. Parsons, I. Rahwan, and C. Reed, Eds., AAMAS 2007, Honolulu, Hawai'i, USA, 15 May 2007, pp. 115–130.

[11] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, no. 2, pp. 321–358, 1995. [Online]. Available: citeseer.ist.psu.edu/dung95acceptability.html

[12] D. N. Walton and E. C. W. Krabbe, *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY Press, 1996.

[13] H. Prakken, "Formal systems for persuasion dialogue," *Knowledge Engineering Review*, vol. 21, no. 2, pp. 163–188, 2006.

[14] ——, "Coherence and flexibility in dialogue games for argumentation," *Journal of Logic and Computation*, vol. 15, no. 6, pp. 1009–1040, 2005.

[15] L. Amgoud, N. Maudet, and S. Parsons, "An argumentation-based semantics for agent communication languages," in *ECAI*, F. van Harmelen, Ed. IOS Press, 2002, pp. 38–42.

[16] S. Parsons, M. Wooldridge, and L. Amgoud, "Properties and Complexity of Some Formal Inter-agent Dialogues," *Journal of Logic and Computation*, vol. 13, no. 3, pp. 347 – 376, 2003.

[17] A. Omicini and F. Zambonelli, "Coordination for Internet application development," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 3, pp. 251–269, Sep. 1999.

[18] E. Oliva, "Argumentation and artifacts for intelligent multi-agent systems," Ph.D. dissertation, Dottorato in Ingegneria Elettronica, Informatica e delle Telecomunicazioni, Cesena, Italy, Mar. 2008.