A Dialogue Game Protocol for Co-operative Plan Proposals

Rolando Medellin-Gasque, Katie Atkinson, and Trevor Bench-Capon

University of Liverpool, Department of Computer Science, Liverpool UK
{medellin,katie,tbc}@liverpool.ac.uk

Abstract. Agents that engage in a dialogue about co-operative plans need to argue about the planning elements in a structured way. We present a Dialogue Game protocol based on an argumentation scheme for plan proposals and associated critical questions that allows agents to engage in dialogues regarding such plans. The syntax of the protocol is presented along with dialogue game rules that allow agents to reach an agreement on the best proposal in a co-operative planning scenario.

Keywords: planning proposal, dialogue game protocols.

1 Introduction

The demands of agent-based computing in uncertain and dynamic environments require planning-agents to communicate and manage their plans resources effectively [19]. In this paper we present work towards using structured argumentation-based dialogues as a way to reach coordination in distributed planning scenarios. Specifically, we present an interaction protocol to handle plan proposals as arguments for action. Each proposals contains a plan that could be questioned and/or attacked according to the rules embedded in the protocol. The protocol provides valid locutions and semantics for agents to exchange their beliefs and values and enable cooperation. The protocol comprises a theory of justification for plan proposals that allows agents to cooperate and agree on the best plan in a scenario. This work is based on dialogue game protocols [1, 2, 13, 14] and arguments over action [3] research.

The remain of the paper is structured as follows, Section 2 presents a general theory to propose plans with an argumentation scheme and the associated critical questions. Section 3 presents the Dialogue Game protocol syntax, rules and semantics. Section 4 describes an implementation of the protocol using Tuple centres and finally in Section 5 we conclude and give future research paths.

2 Plan Proposal scheme and Critical Questions

Artificial Intelligence has become increasingly interested in argumentation schemes due to their potential for making significant improvements in the reasoning capabilities of artificial agents [4] and for automation of agent interactions. In essence,

argumentation is a system for resolving conflicts in terms of the acceptability of the arguments that support the conflicting statements.

Argumentation schemes are stereotypical patterns of defeasible reasoning used in everyday conversational argumentation [23]. In an argumentation scheme, arguments are presented as general inference rules where under a given set of premises a conclusion can be presumptively drawn [24].Our plan proposal *ASP* is as follows:

Given a social context X, in the current circumstances q_0 holding preconditions $\pi(q_0)$, the plan p should be performed to achieve new circumstances q_x , that will hold postconditions $\pi(q_x)$ which will realize the plan-goal G which will promote value(s) V_G.

The valid instantiation of the scheme pre-supposes the existence of a regulatory environment or a social context X in which the proponent-agent has some rights to engage in a dialogue with the cooperating agent. The "social context" was an extension to the argumentation scheme presented in [2] where agents use a social structure to issue valid commands between them. Current circumstances are represented by the initial state q_0. An agent could instantiate the scheme to propose plan p as a finite set of linked action-combinations. The plan leads to a state in which post-conditions $\pi(q_x)$ hold and the plan-goal G is achieved (where G is an assignment of truth values to a set of propositions $p \subseteq \Phi$) and a non-empty set of values is promoted/demoted.

We use Action-based Alternating Transition Systems(AATS) as introduced in [22] as a basis for our formalism to represent action and plan proposals. AATS models define joint-actions that may be performed by agents in a state and the effects of these actions.

Table 1 presents the plan proposal and the AATS model representation.

Table 1. Plan Proposal *ASP*

Plan Proposal	**as an AATS model**
Given a social context X, in the current circumstances q_x holding preconditions $\pi(q_x)$ plan p should be performed to achieve new circumstances q_y that will hold postconditions $\pi(q_y)$ which will realize the plan-goal G which will promote value(s) V_G.	Given social context Δ , In the initial state $q_0 = q_x \in Q$, where $\pi(q_0)$, agents $i, j \in Ag$ should execute plan p, where p is a finite set of joint-actions j_n such that $p = \{j_0, .., j_n\}$ and $\{j_0, .., j_n\} \in J_{Ag}$ and $j_n = \{\alpha_i, .., \alpha_j\}$ with transition given by $\tau(q_x, p)$ is q_y , where $\tau(q_0, \{j_1, .., j_n\}) = \tau(\tau(q_0, j_1), (j_2, ..j_n))$ and $\tau(q_x, \{\}) = q_x$ such that $p_a \in \pi(q_x)$ and $p_a \notin \pi(q_y)$ where $G = p$ and ($V_G \subseteq V$ such that $v_1 \in V_G$ iff $\delta(q_x, q_y, v_1)$ is +) and $V_G \neq \emptyset$

In [25], Walton explains: "*...arguments need to be examined within the context of an ongoing investigation in dialogue in which questions are being asked and answered*". Critical questions are a way to examine the acceptability of arguments instantiating schemes. Depending on the nature of the critical question, they can be used to critique several aspects of the argument.

A benefit of having critical questions associated with an argument scheme is that the questions enable dialogue participants to identify points of challenge in a debate or locate premises in an instantiation of the argument scheme that can be recognized as questionable. A question can be seen as a weak form of attack on a particular element of the argument scheme given different beliefs about the world of the agent posing the question. Critical questions then could be used to create Dialogue Games for agents where the participants put forward arguments (instantiating the argumentation scheme) and opponents to the argument challenge it. Argumentation-based dialogues are used then to formalize dialogues between autonomous agents based on theories of argument exchange. We classify the set of critical questions for the plan proposal scheme in 7 layers.

- Layer 1.- An action and its elements (Lowest level).
- Layer 2.- The timing of a particular action.
- Layer 3.- The way actions are combined.
- Layer 4.- The plan proposal overall.
- Layer 5.- The timing of the plan proposal.
- Layer 6.- Side effects not foreseen.
- Layer 7.- Alternative options (Highest Level).

More details on the proposal scheme could be found in [16] and the complete list of critical questions is given in [15].

3 Dialogue Game Specification

In [26], Walton and Krabbe defined a comprehensive dialogue typology identifying 6 formal models of dialogue: Information Seeking Dialogues, Inquiry Dialogues, Persuasion Dialogues, Negotiation Dialogues, Deliberation Dialogues and Eristic Dialogues (Figure 2).

This categorization is based upon the information at the commencement of the dialogue, in a Persuasion Dialogue for example, the dialogue starts with a proposal and a conflict over it. Examples of communication protocols based on the Walton and Krabbe typology could be found in [1] and [14].

Using this typology to design agent communication is not a straightforward process. First of all, the classification is not exhaustive, an agent dialogue could be from another nature or could be comprised of a combination of several types. In [2] for example, the authors present a protocol that allows agents to engage in dialogues regarding commands, which does not fit into any of the categories by Walton and Krabbe. In our approach, agents are intended to choose the best plan and modify it if necessary, given a conflict in their beliefs and values. Coordination in this scenario may require several types of interactions between

Fig. 1. Determining the type of Dialogue. Image taken from Walton and Krabbe pp.81 [26]

agents. In this protocol we focus on persuasion and deliberation dialogues The main reason to focus on these 2 types follows the assumption that our planning agents, following an internal plan creation, may want to persuade another agents that its' plan is the better option. However, new information may arise and the agent should be able deliberate over a specific action in the plan.

In its simplest variant, a Persuasion Dialogue involves one participant seeking to convince another(s) to accept a statement following a conflict of points of view via a dialectical process [26]. In a typical deliberation dialogue agents conceive proposals for action and move them in the dialogue. If the proposal is favourable for the audience it is accepted and the dialogue move to the next proposal. The course of action for an agent may be selected on the basis of considering preferences or goals. If a proposal is unfavourable agents can reject it or question it [10]. Combination of dialogues (iterated, sequential, parallel, embedded) have been discussed in [11] and [20].

In [13], McBurney and Parsons defined elements that comprise a Dialogue Game specification:

- Commencement Rules: Rules which define the circumstances under which the dialogue could start.
- Locutions: Rules that indicate which utterances are allowed.

– Rules for Combination of Locutions : Rules which define the dialogical context under which particular locutions are permitted or not, or obligatory or not.
– Commitments: Rules which define the circumstances under which participants incur dialogical commitments by their utterances, and thus alter the contents of the participants associated commitment stores.
– Rules for combination of commitments: Rules which define how commitments are combined or manipulated when utterances incurring conflicting or complementary commitments are made.
– Rules for speaker order. Rules which define the order in which speakers may make utterances.
– Termination Rules: Rules that define the circumstances under which the dialogue ends.

We will base our Dialogue Game protocol on these elements, focusing on the locutions and the rules for the combination of them. Our protocol is also based on the Dialogue Game protocol presented in [1] where a Persuasion Dialogue is used to enable agents to argue about proposals for action with a common goal and different preferences.

3.1 Dialogue Game Protocol

Commencement Rules.- The dialogue starts when a valid agent creates the dialogue thread. The validity of the agent should be given by the social context.
Locutions and rules for their combination.- Tables 2 - 6 present the protocol locutions together with their pre-conditions and pos-conditions grouped by dialogue-stages. Dialogue stages present several advantages in terms of the protocol specification and the implementation of it. Our protocol stages are based on the stages presented in [**?**] where dialogue stages for a Deliberation Dialogue are specified as a part of a formal framework for such dialogues [1]. Our dialogue game has six stages:

1. Opening stage.- This stage is to ensure all the agents commit to the cooperate towards agreeing on a plan. The goal could be presented and accepted at this stage.
2. Plan Proposal stage.- This stage is where an agent takes the proponent role and put forward a plan to reach teh goal presented in the previous stage. It is important to mention here that each proposal is evaluated separately, creating an argument tree for each proposal. The protocol enables participants to present several proposals until one is generally accepted.
3. Evaluation stage.- For each proposal, agents acting as respondent could engage in a dialogue comprised of questions and/or attacks over elements presented in the original argumentation scheme.

[1] In [9] Hulstijn uses a similar five-stage model for Negotiation Dialogues.

4. Refinement Stage.-If a plan proposal needs to be refined over certain elements specific locutions enable these tasks. This stage differs from the evaluation stage in the sense that previous proposals are not evaluated (via questioning), but new elements (actions) are presented regarding the original proposal. This is the entry point for the Deliberation Dialogue where agents could engage in a dialogue about what to do given a proposed action in the previous stage is rejected.
5. Acceptance/Rejection stage.- The evaluation of arguments is done in this stage. We consider the dialogue has two outcomes: either a consensus for a plan execution is reached or not. In this stage the accepted proposals could be executed or included in the final plan.
6. Closing stage.- At this stage agents finish their participation in the dialogue following an acceptance or rejection of the proposal

The protocol uses mainly the locutions introduced in the *Fatio* protocol in [12] where the authors extended *FIPA ACL* [7] locutions to handle rational argument dialogues, the locutions are: assert, question, challenge, justify and retract. We add to these locutions the $accept()$, $reject()$, and $retract()$ locutions for a specific proposal or action.

We assume that the language syntax comprises two layers as presented in the FIPA ACL specification . The outer (wrapper) layer comprises the locutions which express the illocutionary force of the inner content (the speech acts) and the inner layer related to the topic of the discussion. We present here the speech acts associated with our Dialogue Game. Tables 2 - 6 present the locutions along with their preconditions and pos-conditions.

Table 2. Syntax Locutions "Opening" Stage

Locution	**Preconditions**	**Postconditions**
$open\_dialogue(d, ag)$	Agent ag is a valid agent	Dialogue d created Dialogue stage: *opening*
$enter\_dialogue(d, ag)$	Dialogue stage *opening* Agent ag not in dialogue	Agent ag in dialogue d
$leave\_dialogue(d, ag)$	Dialogue d created Agent ag in dialogue d	Agent ag out of dialogue Agent ag commitments dropped

Table 3. Syntax Locutions "Proposal" stage

Locution	**Preconditions**	**Postconditions**
$propose\_plan(d, ag, prop_n)$	Dialogue d created Agent ag in Dialogue	Proposal for plan created Agent ag committed to all the elements in $prop_n$ Dialogue stage $proposal_stage$ The $prop_n$ variable includes all the elements regarding the plan proposal.
$leave\_dialogue(d, ag)$	Dialogue d created Agent ag in dialogue d	Agent ag out of dialogue Agent ag Commitments dropped

Table 4. Syntax Locutions Evaluation stage

Locution	**Preconditions**	**Postconditions**
$question(d, ag, prop_n, q_n)$	Proposal $prop_n$ asserted	Element questioned by q_n Dialogue stage: *evaluation* Agent ag takes respondent role
$assert(d, ag, prop_n, \phi, q_n)$	Proposal $prop_n$ open Question q_n presented	Agent ag committed to ϕ the assertion represents an answer for the question q_n
$challenge(d, ag, prop_n, q_n)$	Proposal $prop_n$ open	Element challenged by q_n Agent ag takes respondent role
$justify(d, ag, prop_n, \phi)$	Proposal $prop_n$ open Challenge for φ asserted	Agent ag committed to ϕ
$retract(d, ag, prop_n, \phi)$	Proposal $prop_n$ open Element ϕ asserted by Agent ag	Agent ag not committed to ϕ
$leave\_dialogue(d, ag)$	Dialogue d created Agent ag in d	Agent ag out of dialogue Agent ag commitments dropped

Table 5. Syntax Locutions Refinement Stage

Locution	**Preconditions**	**Postconditions**
$propose\_action(d, ag, prop_n, \alpha)$	Active proposal $prop_n$	Action α asserted
$accept\_action(d, ag, prop_n, \alpha)$	Action α proposed	Action accepted for proposal $prop_n$
$reject\_action(d, ag, prop_n, \alpha)$	Action α proposed	Action rejected for proposal $prop_n$
$retract\_action(d, ag, prop_n, \alpha)$	Action α proposed	Action retracted for proposal $prop_n$
$leave\_dialogue(d, ag, prop_n, \alpha)$	Dialogue d created Agent ag in dialogue d	Agent ag out of dialogue Agent ag commitments dropped

Table 6. Syntax Locutions Acceptance/Rejection Stage

Locution	**Preconditions**	**Postconditions**
$accept\_proposal(d, prop_n)$	Proposal $prop_n$ open	Dialogue stage: *Acceptance_stage* $prop_n$ closed
$reject\_proposal(d, prop_n)$	Proposal $prop_n$ open	Dialogue stage: *Rejection_stage* Proposal $prop_n$ closed
$leave\_dialogue(d, ag)$	Dialogue d created Agent ag in dialogue d	Agent ag out of dialogue Agent ag commitments dropped

Commitments.- Each proposal and assertion presuppose a dialogical commitment for the agent.

Rules for combination of commitments.- Specific locutions are available to drop commitments, contradictory commitments are not allowed.

Rules for the speaker order.- given in the postconditions for each locution.

Termination rules.- The dialogue finishes when a proposal is accepted by all the participants.

4 Dialogue Game Implementation

4.1 *TuCSoN* and *ReSpecT*

To implement our Dialogue Game protocol we use *TuCSoN* (Tuple Centres over the Network), a software platform for *tuple-centre* applications [18]. Tuple-based coordination models originate from the field of paralleling programming, however, their features are useful for coordination of distributed systems [8]. A *tuple-centre* is basically an enhancement of a *tuple space* where agents synchronise and cooperate over information available in a shared data space. A *tuple-centre* is then a *tuple space* enhanced with a behaviour specification that defines the responses (reactions) to communication events [17]. These responses are specified in terms of a reaction specification language. *TuCSoN* uses the *ReSpecT* (Reaction Specification Tuples) language [5] to specify the behaviour of the *tuple-centre*. *ReSpecT* adopts a *tuple language* based on first-order logic to defined logic tuples that are accessible via standard communication operations.

In short, the *TuCSoN* infrastructure works in the following way, the *out* predicate puts a tuple in the *tuple-centre*,while the query primitives (in, rd, inp, rdp) provide tuple templates and expect a matching tuple back from the *tuple-centre*. Specifically *in* and *inp* delete the matching tuple, while *rd* and *rdp* leave it there; *in* and *rd* wait until a suited tuple is available , while *inp* and *rdp* fail if no such tuple is found [17]. Following these behaviour, a *tuple-centre* could be used as a multi-agent coordination platform that provides a data-driven coordination medium through a Dialogue Game protocol plus full observability for agents and selective reactions over communication events.

4.2 Protocol Specification

The protocol locutions and rules are pre-loaded into the *tuple-centre*. The *TuCSoN* infrastructure allow us to save *tuple-centres* making them persistent. Valid locutions are in the form of persistent tuples and protocol rules are specified using *ReSpecT* reactions. Intuitively, agents should not be allowed to remove locution tuples from the *tuple-centre*. This feature assumes a read-only permissions for agents to remove specific tuples (we will not a address this feature here).

Table 7 present the *ReSpecT* tuples used to define the syntax of our protocol. Variables used in the locutions stands for

- Di: dialogue identifier.
- Ag: Agent identifier.
- Pr: Proposal identifier
- Ac: Action identifier.
- CQ: Critical question number.
- Ev: evidence.

Table 7. *ReSpecT* Dialogue Game Locutions

Locution	***ReSpecT* format**
$open\_dialogue()$	$loc(open(Di, Ag))$
$enter\_dialogue()$	$loc(enter(Di, Ag))$
$propose()$	$loc(propose(Di, Ag, Pr))$
$accept\_proposal()$	$loc(accept\_proposal(Di, Ag, Pr))$
$reject\_proposal()$	$loc(reject\_proposal(Di, Ag, Pr))$
$retract\_proposal()$	$loc(retract\_proposal(Di, Ag, Pr))$
$propose\_action()$	$loc(propose\_action(Di, Ag, Pr, Ac))$
$accept\_action()$	$loc(accept\_action(Di, Ag, Pr, Ac))$
$reject\_action()$	$loc(reject\_action(Di, Ag, Pr, Ac))$
$retract\_action()$	$loc(retract\_action(Di, Ag, Pr, Ac))$
$question()$	$loc(question(Di, Ag, Pr, CQ))$
$assert()$	$loc(assert(Di, Ag, Pr, CQ, E))$
$challenge()$	$loc(challenge(Di, Ag, Pr, CQ))$
$justify()$	$loc(justify(Di, Ag, Pr, CQ, E))$
$leave\_dialogue()$	$loc(leave\_dialogue(Di, Ag))$

Using *ReSpecT* it is possible to define rules that capture how a dialogue state changes when a locution is made. Urovi *et.al* [21] use automated work-flows with interactions governed by dialogue game implemented in *TuCSoN*. Our protocol then, could be specified as a a set of rules embedded in the *tuple-centre* represented as reactions to locution-inputs from agents in the following way:

$$reaction(out(Locution1()), \text{when } Conditions\_n \text{ then } out\_r(Dialogue(Locution1)$$

where for every $Locution1$ inserted in the tuple-centre a set of conditions are checked (*e.g.* Is it a valid locution? Does the actual stage allows the locution?) , if the conditions are correct consequences are applied , (*e.g.* the dialogue history is updated, the dialogue stage changes, the commitment store is updated, etc).

Table 8 presents the *ReSpecT* reaction to the *enter_dialogue* locution and table 9 the reaction to the "propose" locution. Figure **??** presents a diagram with the process each time a tuple is inserted in the tuple center.

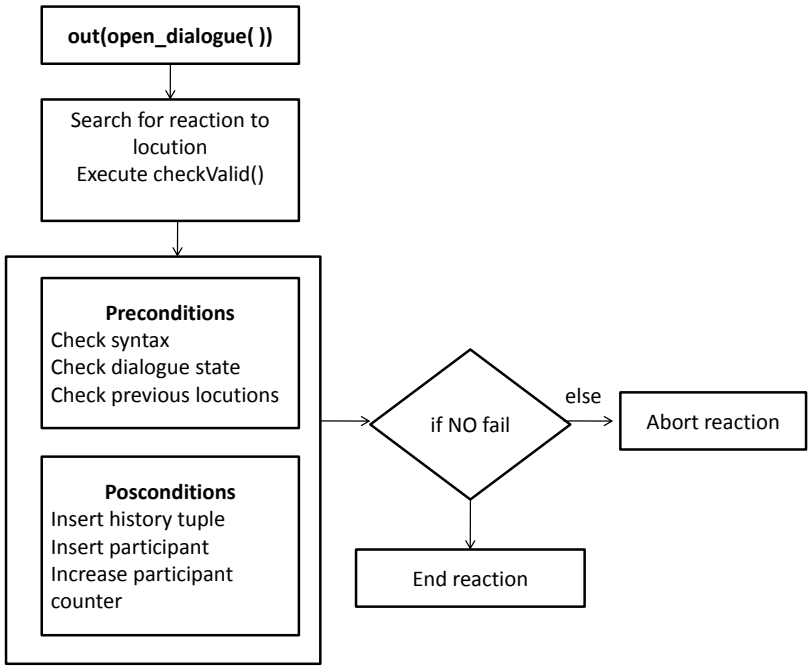

Fig. 2. TupleCentre Reaction to the open locution

The critical questions are specified with the following structure:

$cQ(layer1, qA01, Action, valid)$ where $cQ01$ is the critical question number, in later 1, which questions the validity of the action specified. In this way all the questions are specified in the tuple centre determining the way in which the proposal could be questioned and/or attacked.

4.3 Using the protocol

To interact with the protocol we implemented JAVA-agents that use the TuCSoN framework classes. The agents post tuples and the tuple center reacts accordingly. So if a not valid locution is posted the tuple centre does not react to the event. the way to coordinate agents using the protocol is to insert tuples to wait for an specific event. for example, when an agent input a proposal in the same process a tuple to read questions related to that proposal is inserted. So if an agent questions an element of the proposal the proponent agent reads it automatically and process it.

Known approaches to implement Dialogue protocols could be found in [6] and [1]. In [6] Doutre *et. al.* implement an Information-Seeking Dialogue focused on permissions, the coordination mechanism is implemented in *TuCSoN*. In [1] Atkinson *et. al.* implements a Persuasion Dialogue using Java elements. The main

Table 8. *ReSpecT* Semantics for the the *enter_dialogue* locution

***ReSpecT* reaction**	**Description**
$reaction(out(enter_dialogue(Di, Ag)),$ $(out_r(checkValid(loc(enter_dialogue(Di, Ag))))$	If enter() tuple is inserted check locution
$reaction(out\_r($ $checkValid(loc(enter\_dialogue(Di, Ag)))),$	internal reaction checkValid,
Preconditions	
$rd\_r(loc(enter\_dialogue(\_, \_))),$	if tuple is in the syntax
$rd\_r(dState(open)),$	check dialogue state
$no\_r(dhistory(enter\_dialogue(\_, Ag))),$	check previous locution by same agent
$no\_r(dhistory(open\_dialogue(\_, Ag))),$	the dialogue should not be opened by the same agent
Postconditions	
$out\_r(participant(Di, Ag)),$	register participant
$in\_r(n\_participants(N)),$	increase the participants counter
$N1$ is $N + 1,$	
$out\_r(n\_participants(N1)),$	
$out\_r(dhistory(enter\_dialogue(Di, Ag))),$	insert dHistory tuple
$in\_r(checkValid((\_))),$	clean auxiliary tuples
$in\_r(enter\_dialogue(\_, \_))))$	

difference is that our protocol syntax and semantics are completely embedded in the *tuple-centre*.

5 Conclusions and Future work

This paper presents the syntax and semantics for a agent dialogue game protocol for argument over co-operative plans together with a description of the implementation. This paper has two main contributions: first, a novel dialogue protocol is defined to handle several proposals under a single dialogue with the semantics completely embedded in the tuple center. Another contribution is that the protocol allows to change the dialogue type at a specific point.Future work involves implementing agents using the protocol and using predefined strategies to select relevant questions to maximize the cooperation between agents.

References

1. K. Atkinson, T. Bench-Capon, and P. McBurney. A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems*, 11:153–171, September 2005.
2. K. Atkinson, R. Girle, P. McBurney, and S. Parsons. Command Dialogues. In I. Rahwan and P. Moraitis, editors, *Argumentation in Multi-Agent Systems*, Fifth International Workshop, pages 93–106, Berlin, Heidelberg, 2009. Springer-Verlag.
3. T. Bench-Capon and K. Atkinson. Action-state Semantics for Practical Reasoning. *AAAI Fall Symposium. Technical Report SS-09-06, AAAI Press*, pages 58–63, 2009.

Table 9. *ReSpecT* Semantics for the *propose_dialogue* locution

***ReSpecT* reaction**	**Description**
$reaction(out(propose\_plan(Di, Ag, Pr)),$ $(out\_r(checkValid$ $(propose\_plan(Di, Ag, Pr))))).$	specify the tuple to react specify the tuple to react
$reaction($ $out\_r(checkValid(propose\_plan(Di, Ag, Pr))),$	
Preconditions	
$(rd\_r(loc(propose\_plan(\_, \_, \_))),$	verify the locution is in the syntax
$rd\_r(dState(open)),$	check dialogue state
$rd\_r(participant(Di, Ag)),$	confirm participant is in the dialogue
$no\_r(dhistory(propose\_plan(\_, Ag, \_))),$	check previous locution by same agent
Postconditions	
$out\_r(dhistory(propose\_plan(Di, Ag, Pr))),$	insert dHistory tuple
$out\_r(dState(proposing)),$	change state
$in\_r(dState(open)),$	delete previous state
$out\_r(role(Di, proponent, Ag)),$	assign a dialogue role to the agent
$in\_r(checkValid((\_))),$	clean auxiliary tuples
$in\_r(propose_plan(\_, \_, \_)))).$	

4. F. Bex, H. Prakken, C. Reed, and Douglas Walton. Towards a formal account of reasoning about evidence: argumentation schemes and generalisations. *Artificial Intelligence Law*, 11(2-3):125–165, 2003.
5. E. Denti, A. Natali, and A. Omicini. On the Expressive Power of a Language for Programming Coordination Media. In *In Proceedings of the 1998 ACM Symposium on Applied Computing (SAC98)*, pages 169–177, 1998.
6. S Doutre, P. McBurney, M. Wooldridge, and W. Barden. Information-seeking agent dialogs with permissions and arguments. Technical Report ULCS-05-010, Department of Computer Science. University of Liverpool, Liverpool UIK, 2005.
7. FIPA. Communicative Act Library Specification. Standard SC00037J, Foundation for Intelligent Physical Agents, 3 December 2002.
8. D. Gelernter and N. Carriero. Coordination languages and their significance. *Commun. ACM*, 35, February 1992.
9. J. Hulstijn. *Dialogue Models for Inquiry and Transaction*. PhD thesis, Universiteit Twente, Enschede, The Netherlands, 2000.
10. P. McBurney, D. Hitchcock, and S. Parsons. The Eightfold Way of Deliberation Dialogue. *International Journal of Intelligent Systems*, 22(1):95–132, 2007.
11. P. McBurney and S. Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 11:2002, 2001.
12. P. McBurney and S. Parsons. Syntax and Semantics of the Fatio Argumentation Protocol. Technical Report ULCS-04-002, University of Liverpool, 2004.
13. P. McBurney and S. Parsons. Dialogue Games for Agent Argumentation. In I. Rahwan and G. Simari, editors, *Argumentation in Artificial Intelligence*, chapter 13, pages 261–280. Springer, Berlin, Germany, 2009.
14. P. McBurney, R. M. Van Eijk, S. Parsons, and L. Amgoud. A Dialogue Game Protocol for Agent Purchase Negotiations. *Autonomous Agents and Multi-Agent Systems*, 7:235–273, 2003. 10.1023/A:1024787301515.

15. R. Medellin-Gasque, K. Atkinson, and T. Bench-Capon. An Analys of Critical Questions for Co-operative Plan Proposals. Technical report, Department of Computer Science, University of Liverpool, UK, September 2011.
16. R. Medellin-Gasque, K. Atkinson, P.McBurney, and T. Bench-Capon. Arguments over co-operative plans. In *In: Proceedings of the First International Workshop on Theory and Applications of Formal Argumentation (TAFA 2011)*, Barcelona Spain, September 2011.
17. A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277 – 294, 2001.
18. A. Omicini and F. Zambonelli. Coordination for internet application development. *Autonomous Agents and Multi-Agent Systems*, 2:251–269, September 1999.
19. M. E. Pollack and J. F. Horty. There's more to life than making plans: Plan management in dynamic, multi-agent environments. *AI Magazine*, 20:20–4, 1999.
20. Chris Reed. Dialogue Frames in Agent Communication. In *In Proceedings of the Third International Conference on Multi-Agent Systems*, pages 246–253. IEEE Press, 1998.
21. V. Urovi, S. Bromuri, J. Mcginnis, K. Stathis, and A. Omicini. Automating Workflows Using Dialetical Argumentation. *IADIS International Journal on Computer Science and Information System*, 3(2):110–125, 2008.
22. W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility, and synthesis. *Synthese*, 156:1–19, 2007.
23. D. N. Walton. What is Reasoning? What is an Argument? *Journal of Philosophy*, 87:399–419, 1990.
24. D. N. Walton. *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1996.
25. D. N. Walton. Justification of argumentation schemes. *Australasian Journal of Logic*, 3, 2005.
26. D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY Series in Logic and Language. State University of New York Press, Albany, NY, USA, 1995.