A Generative Dialogue System for Arguing about Plans in Situation Calculus

Alexandros Belesiotis[1], Michael Rovatsos[1], and Iyad Rahwan[2]

[1] School of Informatics, The University of Edinburgh,
Edinburgh EH8 9LE, UK
{A.Belesiotis,Michael.Rovatsos}@ed.ac.uk
[2] Institute of Informatics, The British University in Dubai,
P.O. Box 502216, Dubai, UAE
irahwan@acm.org

Abstract. This paper presents an argumentation mechanism for reconciling conflicts between planning agents related to plan proposals, which are caused by inconsistencies between basic beliefs regarding the state of the world or the specification of the planning operators.
We introduce simple and efficient argument moves that enable discussion about planning steps, and show how these can be integrated into an existing protocol for belief argumentation. The resulting protocol is provably sound with regard to the defeasible semantics of the resulting agreements. We show how argument generation can be treated, for the specific task of argumentation about plans, by replacing the burden of finding proofs in a knowledge base by guided search.

1 Introduction

In recent years, argumentation [1] has attracted much attention as a technique for resolving conflicts between agents, mainly due to its strong logical foundation and its suitability for use in multiagent situations.

One area in which argumentation has been recently employed is that of collaborative practical decision making [2–4], e.g. when deciding on task allocation in teamwork frameworks [5]. In this case, it can be imagined that different agents come up with proposals for joint action, such as multiagent plans, and discuss such plans with their teammates in order to reach agreement. The need for conflict resolution may arise from the agents having different viewpoints due to locality of sensing, different fundamental assumptions about the domain, or simply because different agents may have conducted different inferences and therefore their beliefs may not be aligned.

In cooperative distributed problem solving [6] (e.g. frameworks like GPGP [7]), conflicts are normally resolved by merging agents' different views. However, this is only feasible if agents use highly structured knowledge bases that contain only knowledge relevant to the task in hand. In more general agent designs (where

agents have arbitrary beliefs about the world), detecting conflicts when merging beliefs is computationally complex and, in some cases, completely unnecessary and wasteful. Efficiency can be increased by identifying conflicts that are related to plans that are currently being debated, and resolving only disagreements that affect the viability of concrete plan proposals.

In this paper, we propose a method that aims to tackle precisely this problem. Based on the observation that logical theories of planning are highly structured and fairly simple, we devise an argumentation protocol that allows agents to discuss plan proposals and to identify reasons for potential disagreements that originate in differences regarding beliefs about the planning domain. We provide an algorithm and explain how the proposed protocol can be used to guide the search for relevant disagreements in a focused way. We also show that it is easy to integrate our argument moves with conventional belief argumentation protocols, and discuss useful properties of the resulting framework for reasoning about plans. An overview of the process is described in Figure 1. In this paper we do not focus on the process of planning itself, but on how structured argumentation-based dialogue can be employed as the means for identification of disagreements regarding plans.

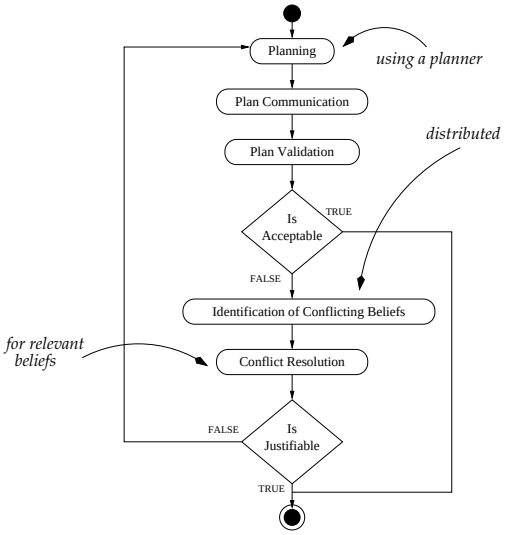

Fig. 1. Outline of how the employed protocol can be used to identify and resolve conflicts about plans

The paper advances the state-of-the-art in multiagent planning in two ways. Firstly, we present the first dialogue protocol that is specifically designed to enable agents to detect and resolve disagreements about a plan generated via conventional planning algorithms. This contrasts with other approaches [8, 3] which are concerned with the process of planning itself. The second contribution of this paper is to show that the dialogue generation process can be simplified

by exploiting the structure of the planning problem to systematically discover disagreements over only the beliefs relevant to the plan structure. In particular, in contrast to general protocols [2] which allow very expressive dialogues about goals, intentions, etc, but come with no clear bounds on the length of dialogues, we give exact upper bounds on the number of messages exchanged before disagreements are identified.

The remainder of this paper is structured as follows: Section 2 introduces the argumentation and planning background required for the development of our framework. Section 3 introduces the suggested protocol, followed by an example illustrating the protocol. Section 5 provides an analysis of protocol properties, and in Section 6 we present an algorithm for argument generation. Section 7 concludes.

2 Preliminaries

2.1 Planning framework

Our planning framework follows the *situation calculus* framework [9], a logical language designed for representing dynamic domains. Three disjoint sorts are supported. The sort *action* represents actions, the sort *situation* represents situations and the sort *object* all the rest. S_0 is a constant symbol representing the initial situation. The binary function symbol $do : action \times situation \rightarrow situation$ denotes the successor situation after preforming an action. $Poss : action \times situation$ is a binary predicate symbol representing whether an action is applicable in a situation. The binary predicate symbol $\sqsubset$: $situation \times situation$ defines an ordering relation over situations, where $s \sqsubset s'$ denotes that s is a proper subsequence of s'. Symbols whose value change in different situations are called fluents (relational or functional), and they have an argument of sort situation as their final argument.

In particular, the dialogue attempts to identify and reconcile beliefs in the agents' theories about the world that conflict. Each agent maintains the representation of the domain as a *Basic Action Theory* $\mathcal{D}$, as proposed by Reiter in [10].

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$$

Where Σ is a set of Fundamental domain-independent axioms providing the basic properties for situations. Successor state axioms $\mathcal{D}_{ss}$ are introduced for each relational or functional fluent in the domain, and specify the conditions that govern its value in a situation. The conditions under which an action can be performed are specified by the action precondition axioms, $\mathcal{D}_{ap}$. $\mathcal{D}_{S_0}$ is a set of first-order sentences that represent the initial state of the world. Each Successor State axiom (A_{ss}) describes the conditions that should hold in situation s so that a relational or functional fluent takes a certain value in the situation (a, s), which follows from the application of action a in s. For relational and functional fluents respectively, they have the following form:

$$F(x_1, ..., x_n, do(a, s)) \equiv \Phi_F(x_1, ..., x_n, a, s)$$

$$f(x_1, ..., x_n, do(a, s)) = y \equiv \Phi_f(x_1, ..., x_n, y, a, s)$$

Action Precondition axioms (A_{ap}) specify the preconditions of actions and have the following form:

$$Poss(A(x_1, ..., x_n), s) \equiv \Pi_A(x_1, ..., x_n, s)$$

Situation calculus treats a plan, for a goal sentence G relative to a domain $\mathcal{D}$, as a variable-free situation term s_π iff $\mathcal{D} \vDash executable(s_\pi) \wedge G(s_\pi)$, where $executable(s_\pi) \stackrel{\text{def}}{=} (\forall a, s^*).do(a, s^*) \sqsubseteq s \supset Poss(a, s^*)$.

Definition 1. *A* planning problem $\mathcal{P}$ *is a tuple* $\langle \mathcal{D}, G \rangle$*, where* $\mathcal{D}$ *is the planning domain and* G *describes a fluent symbol specifying the goal.*

A consequence of the definition of a plan in situation calculus and the foundational axioms for situations is that $executable(do(a, s)) \equiv executable(s) \wedge Poss(a, s)$. This enables the definition of a plan as a sequence of actions solving a planning problem.

Definition 2. *A plan* $\pi = a_1; a_2; \ldots; a_n$ *is a* solution *to a planning problem* $\mathcal{P}$ *iff* $\mathcal{D} \vDash Poss(a_1, S_0) \wedge do(a_1, S_0) = s_1 \wedge Poss(a_2, s_1) \wedge do(a_2, s_1) = s_2 \wedge \ldots \wedge Poss(a_n, s_{n-1}) \wedge do(a_n, s_{n-1}) = s_n \wedge G(s_n)$.

This definition describes that each action in the plan can be performed in the situation that results from the application of the previous action in the sequence, and that after preforming the final action the goal sentence G will be true.

In classical AI planning, domain knowledge is assumed to be conflict-free. In a multiagent setting, however, there are cases where agents have mutually conflicting beliefs. In order to find a plan that satisfies all parties, the agents need to solve a planning problem involving conflicting knowledge about the planning domain.

Assume two agents i and $-i$ each of which has an (internally consistent) domain knowledge $\mathcal{D}_i/\mathcal{D}_{-i}$. A planning problem $\mathcal{P}$ is set to be *potentially conflicted* among agents i and $-i$, if there exist statements in $\mathcal{D}_i$ that conflict with statements in $\mathcal{D}_{-i}$.[3]

Assuming that the goal has been agreed upon, then due to the contradictory beliefs of the agents, it can be challenging to discover a plan that satisfies both i and $-i$, since from each agent's local perspective, the plan needs to solve a different planning problem: i is looking for a solution to $\mathcal{P}_i = \langle \mathcal{D}_i, G \rangle$, whereas agent $-i$ wants to solve $\mathcal{P}_{-i} = \langle \mathcal{D}_{-i}, G \rangle$. An *acceptable plan* that can be agreed upon by two agents i and $-i$ is therefore defined as follows:

Definition 3. *Given the domain representations for the two agents* $\mathcal{D}_i$ *and* $\mathcal{D}_{-i}$*, and a common goal* G*, a* plan is acceptable *if and only if it is the solution to both planning problems* $\mathcal{P}_i = \langle \mathcal{D}_i, G \rangle$ *and* $\mathcal{P}_{-i} = \langle \mathcal{D}_{-i}, G \rangle$.

With this, the purpose of our argumentation protocol can be specified as follows: *for a potentially conflicted planning problem* $\mathcal{P}$ *between two agents and an initial proposed plan* π *determine whether this plan is acceptable.*

[3] This paper assumes two-player situations; in case of more than two agents, our results carry over assuming dialogues are conducted between all pairs to reach agreement.

2.2 Argumentation Framework

Argumentation is a mechanism that can be used for the resolution of conflicts. Our basic argumentation framework follows [1]:

Definition 4. *An* argumentation system *is a structure* $\mathcal{H} = \langle \mathcal{A}, \rightarrow \rangle$ *where* $\mathcal{A}$ *is a set of arguments and* $\rightarrow \subseteq \mathcal{A} \times \mathcal{A}$ *is a binary* attack *relation between arguments. The following additional definitions on sets of arguments are useful:*

1. *A set of arguments* $S \subseteq \mathcal{A}$ *is* conflict-free *iff* $(\nexists A, B \in S).A \rightarrow B$
2. *An argument* $A \in \mathcal{A}$ *is* acceptable with respect to a set $S \subseteq \mathcal{A}$ *of arguments iff* $(\forall B \in \mathcal{A}).(B \rightarrow A \Rightarrow [(\exists C \in S).C \rightarrow B])$
3. *A set of arguments* $S \subseteq \mathcal{A}$ *is called* admissible *if it is conflict-free and if each argument in* S *is acceptable with respect to* S
4. *A* preferred extension *of an argumentation framework is a maximal (with respect to set inclusion) admissible set of the argumentation framework.*
5. S *is a* stable extension *of an argumentation framework, if* S *is conflict-free and* $(\forall A \notin S).[\exists B \in S).(B \rightarrow A)]$

Further refinements that we use below include (i) *credulous preferred semantics*, which require an argument to be part of at least one preferred extension to be considered acceptable, and (ii) *sceptical preferred semantics*, which require that the argument is contained in all preferred extensions of the argumentation framework.

To apply these notions in a logic-based framework, we define arguments based on an inference procedure $\vdash$ in a knowledge base, and reinterpret the attack relation using logical contradiction:

Definition 5. Arguments *for agent* i *are pairs* $A = \langle H, h \rangle$*, where* $H \subseteq \mathcal{D}_i$*, and*

i. H *is consistent (i.e.* $H \nvdash \bot$*),*
ii. $H \vdash h$*,*
iii. H *is minimal (no subset of* H *satisfies both* i. *and* ii.*).*

H *is called the support of the argument and* h *its conclusion.*

If an argument follows the aforementioned definition then we can refer to it as being a *valid* argument.

Definition 6. *An argument* $A_1 = \langle H_1, h_1 \rangle$ attacks *an argument* $A_2 = \langle H_2, h_2 \rangle$*, denoted* $A_1 \rightarrow A_2$*, if* $\exists \phi$ *in* H_2 *such that* $h_1 \equiv \neg \phi$*.*

3 Dialogue protocol

We suggest a protocol that is a *two-party immediate response dispute* (TPI-dispute) type protocol and builds on [11, 12]. It extends the protocol presented in [11] to include planning-related argument moves (for proposing a plan, attacking a plan and attacking the attackers of a plan).

In the protocol described in [11], the dialogue moves COUNTER, BACKUP and RETRACT may be employed to progress the dialogue by attacking the other party's most recent argument, or by returning to an earlier point in the

dialogue and providing an alternative attack. Rules are provided formulating the conditions under which the moves can be used, depending on the "current" state of the dispute and the arguments that are available to each agent. We have adapted this protocol so that the applicability of moves can be evaluated only with respect to the state of the dialogue. This is necessary for our system as agents do not know all the possible arguments that can be created from their beliefs. Instead they construct candidate responses before making their next move.

Planning-related moves are provided as instantiations of TPI-dispute moves, through further restrictions on the preconditions and the form of the exchanged argument. These restrictions also depend on the state of the dialogue.

3.1 TPI-Disputes

In order to discover if an argument to follow a plan is acceptable (and therefore if a plan is acceptable), the agents engage in an argumentation game. The agent proposing the plan, initiates the game, and plays the role of the proponent PRO, leaving the role of opponent OPP to the other party. The proponent agent is responsible for constructing arguments in favour of the plan, while the opponent is attempting to attack the validity of the plan. The game progresses with the agents exchanging arguments attacking the previous argument of their rival. The proponent attempts to create an admissible set containing the initial argument, which in this case proposes a plan believed to achieve the shared objective. The following definitions follow [11].

Definition 7. *Let $\mathcal{H} = \langle \mathcal{A}, \rightarrow \rangle$ be an argumentation system with $\mathcal{A} = \mathcal{A}_{PRO} \cup \mathcal{A}_{OPP}$ the union of the arguments that can be constructed by the proponent and the opponent. A* dispute tree for some argument X in $\mathcal{A}_{PRO}$*, denoted by $\mathcal{T}_A^{\mathcal{H}}$, is a tree with root X whose vertices and edges are subsets of $\mathcal{A}$ and $\rightarrow$, respectively.*

Note that the edges in a dispute tree are directed from vertices to their parent node. A *dispute line* is a path in the dispute tree, and is of the form

$$t = v_k \rightarrow ... \rightarrow v_1 \rightarrow v_0 = X$$

A dispute line is called *open/closed* if the agent who has to make the following move is able/unable to construct an argument following Definition 5. A closed dispute line for some argument X is a *failing defence of* X if the leaf node argument move has been made by the opponent. On the contrary, if the final argument has been raised by the proponent the dispute line is considered a *failing attack of* X.

A TPI-dispute for some argument X of a system $\mathcal{H}$ is a sequence of moves

$$M = \langle \mu_1, \mu_2, ..., \mu_i, ... \rangle$$

The agents have a finite repertoire of available move types. The moves available to the proponent are COUNTER_{PRO} and RETRACT_{PRO}, whereas the moves available to the opponent are COUNTER_{OPP} and BACKUP_{OPP}. Whether it is

possible for an agent to perform a move is decided depending on move preconditions and the current state of the dispute. The state of the dispute after the kth move, with $(k \geq 0)$, is described by the following tuple:

$$\sigma_k = \langle T_k, v_k, CS_k^{PRO}, CS_k^{OPP}, P_k, Q_k \rangle$$

T_k is the dispute tree after move k. The last argument proposed is denoted by v_k. CS_k^{PRO}/CS_k^{OPP} contain the proponent's/opponent's commitments. P_k contains arguments that are presented as a subset of the admissible set, and Q_k contains sets that have been shown not to be a subset of an admissible set. The initial state of the dispute for some argument X is

$$\sigma_0 = \langle \langle X \rangle, X, \{X\}, \{\}, \{X\}, \emptyset \rangle.$$

Turntaking in the dialogue is determined by the running index of the current move. The current player is OPP if the index of the current move k is odd, and PRO otherwise. If the current player has a legal move available, then the dispute is active; if not, then it terminates. When a dialogue terminates, the winner can be determined using the number of moves, $|M|$. If $|M|$ is odd/even, then the proponent/opponent wins the argument, respectively.

The repertoire of moves available to the agents and their applicability will be presented in terms of preconditions and effects related to the state of the dispute before the moves are applied.

If k is odd, the opponent may attack the most recent argument presented by the proponent by putting forward argument Y, using the $\mu_k = COUNTER_k^{OPP}(Y)$ move.

$$\boxed{\mu_k = COUNTER_k^{OPP}(Y)}$$

Preconditions:
- Y is a valid argument;
- $Y \rightarrow v_{k-1}$;
- $Y \notin CS_{k-1}^{OPP}$;
- $Y \notin CS_{k-1}^{PRO}$;
- $(\nexists\ Z \in CS_{k-1}^{PRO}).\ Z \rightarrow Y$.

Effects:
- $T_k := T_{k-1} + \langle Y, v_{k-1} \rangle$;
- $v_k := Y$;
- $CS_k^{PRO} := CS_{k-1}^{PRO}$;
- $CS_k^{OPP} := CS_{k-1}^{OPP} \cup Y$;
- $P_k := P_{k-1}$;
- $Q_k := Q_{k-1}$.

The definition of this move describes that in order to respond to the most recent argument put forward by the proponent, the opponent must construct a valid argument, according to Definition 5, that attacks the proponent's previous argument v_{k-1}. The opponent cannot reuse arguments that have already been presented by her or by the proponent, or any arguments that can be attacked by arguments previously proposed by the proponent. This dialogue move affects the state of the dialogue by including the new argument in the dispute tree as the most recently presented argument, as well as in the opponent's commitments.

A variation of the counter available to the proponent is defined in a similar fashion.

$$\mu_k = COUNTER_k^{PRO}(Y)$$

Preconditions:
- Y is a valid argument;
- $Y \rightarrow v_{k-1}$;
- $Y \notin CS_{k-1}^{PRO}$;
- $(\nexists\ Z \in CS_{k-1}^{PRO}).\ Z \rightarrow Y$.

Effects:
- $T_k := T_{k-1} + \langle Y, v_{k-1} \rangle$;
- $v_k := Y$;
- $CS_k^{PRO} := CS_{k-1}^{PRO} \cup Y$;
- $CS_k^{OPP} := CS_{k-1}^{OPP}$;
- $P_k := P_{k-1} \cup Y$;
- $Q_k := Q_{k-1}$.

The proponent, when using the counter move, cannot repeat arguments or state arguments that contradict her commitments. This move progresses the dispute by adding Y to the dispute tree, the proponent's commitments and to the subset of the admissible set that the proponent is attempting to construct.

The backup move can be used by the opponent if a counter move responding to the proponent's most recent argument is not possible. This move returns to the most recent point in the dispute, in which the opponent can proceed with an alternative attack. This point is represented by j, which must be even and $0 \leq j \leq k-3$.

$$\mu_k = BACKUP_k^{OPP}(j, Y)$$

Preconditions:
- Cannot construct a valid argument that attacks v_{k-1};
- For each r in $\{j+2, j+4, \ldots, k-3\}$, no argument Y' exists that
  - Y' is a valid argument;
  - $Y' \rightarrow v_r$;
  - $Y' \notin CS_r^{OPP} \cup \{v_{r+1}, v_{r+3}, \ldots, v_{k-2}\}$;
  - $Y' \notin CS_r^{PRO} \cup \{v_{r+2}, v_{r+4} \ldots, v_{k-3}\}$;
  - $(\nexists\ Z \in CS_r^{PRO} \cup \{v_r, v_{r+2}, \ldots, v_{k-3}\}).\ Z \rightarrow Y'$.
- Y is a valid argument;
- $Y \rightarrow v_j$;
- $Y \notin CS_j^{OPP} \cup \{v_{j+1}, v_{j+3}, \ldots, v_{k-2}\}$;
- $Y \notin CS_j^{PRO} \cup \{v_{j+2}, v_{j+4} \ldots, v_{k-3}\}$;
- $(\nexists\ Z \in CS_j^{PRO} \cup \{v_j, v_{j+2}, \ldots, v_{k-3}\}).\ Z \rightarrow Y$.

Effects:
- $T_k := T_{k-1} + \langle Y, v_j \rangle$;
- $v_k := Y$;
- $CS_k^{PRO} := CS_{k-1}^{PRO}$;
- $CS_k^{OPP} := CS_j^{OPP} \cup Y \cup \{v_{j+1}, v_{j+3}, \ldots, v_{k-2}\}$;
- $P_k := P_{k-1}$;
- $Q_k := Q_{k-1}$.

This move can be used by the opponent to attack the argument presented by the proponent in move μ_j, if there do not exist any more recent points in which the opponent can mount an alternative attack. The argument Y must not have been repeated by the proponent or the opponent, and it must not be attacked by any arguments presented by the proponent.

The following move can be used by the proponent in order to retract and provide an alternative justification for X, if the opponent has shown that P_k is not part of an admissible set. By retracting, PRO attempts to construct a different admissible set containing X.

$\mu_k = RETRACT_k^{PRO}$

Preconditions:
- PRO cannot attack u_{k-1};
- $P_k \neq \{X\}$.

Effects:
- $T_k := \langle X \rangle$;
- $v_k := X$;
- $CS_k^{PRO} := CS_0^{PRO}$;
- $CS_k^{OPP} := CS_0^{OPP}$;
- $P_k := P_0$;
- $Q_k := Q_{k-1} \cup \{P_{k-1}\}$.

According to [11], if there exists a terminated TPI-dispute over an argument X in the argument system $\mathcal{H}$ that is a successful defence of X (i.e. k is even), then at least one preferred extension of $\mathcal{A}$ contains X, and X is said to be credulously accepted in $\mathcal{H}$.

To define sceptical acceptance semantics, Dunne and Bench-Capon [11] use the notion of an *x-augmented system* $\mathcal{H}_a$ which is formed for an argument system $\mathcal{H}(\mathcal{A}, \rightarrow)$ and $X \in \mathcal{A}$, by introducing a new argument X_a and an attack $\langle X, X_a \rangle$. For an argument system $\mathcal{H}(\mathcal{A}, \rightarrow)$, in which every preferred extension is a stable extension, an argument $X \in \mathcal{A}$ is sceptically accepted (i.e. is part of every preferred extension) in $\mathcal{H}$, if and only if there is a dispute M, providing a successfull rebuttal of X_a in the x-augmented system $\mathcal{H}_a$.

3.2 Arguments in Situation Calculus

Dialogue about plans may be broken down to two distinct levels. The *Plan Level Dispute* (PLD) involves arguments regarding future states of the world, explaining the agents' views on how the proposed plan will affect the environment. The *Domain Level Dispute* (DLD) involves arguments about the planning domain, and involve beliefs about the initial state of the world or the specification of the planning operators. All PLD and DLD arguments need to follow Definition 5 and attack the most recent argument proposed by the other party, according to the preconditions of the dialogue moves. In order for the following argument types to be instantiated into arguments respecting Definition 5, the conditions for using each argument type need to be followed.

Plan Level Dispute. The argument game is initiated through a plan *Proposal* argument. The agent that introduces this argument plays the role of the proponent. The proposed plan is considered to be acceptable with respect to the proponent's beliefs:

Proposal (P), for $\pi = a_1; a_2; \ldots; a_n$

- *Argument:*
  $\{Poss(a_1, S_0), s_1 = do(a_1, s_0), \ldots, Poss(a_n, s_{n-1}), s_n = do(a_n, s_{n-1}), G(s_n)\} \vdash$
  $Poss(a_1, S_0) \wedge s_1 = do(a_1, s_0) \wedge \ldots \wedge Poss(a_n, s_{n-1}) \wedge s_n = do(a_n, s_{n-1}) \wedge G(s_n)$

Invalid Action arguments show that a certain action in the plan cannot be applied in the relevant situation. The support of such arguments states the relevant

Action Precondition axiom, and a statement about the situation in which the action should be performed $\neg \Pi_a(s)$ describing that the conditions that make the action applicable are not the case in s.

Invalid Action (IA)

- *Preconditions:*
  - $Poss(a, s) \in H_{v_{k-1}}$, where $H_{v_{k-1}}$ contains the support for v_{k-1}
- *Argument:*
  $\{\neg \Pi_a(s), Poss(a(x_1, \ldots, x_n), s) \equiv \Pi_a(x_1, x_2 \ldots, x_n, s)\} \vdash \neg Poss(a, s)$

Statement does Not Hold in s arguments can be used to attack statements $\Phi(s)$ regarding a future situation. They explain that a series of (relational or functional fluent and non-fluent) terms which appear in $\Phi(s)$, take values that ensure that $\Phi(s)$ is not the case. Such arguments use the relevant successor state axiom of every fluent symbol for the support, in order to describe the conditions in the previous situation that led the specific fluents to take certain values that make the statement $\Phi(s)$ to be false.

Statement does Not Hold in s (NHS)

- *Preconditions:*
  - $\Phi(s) \in H_{v_{k-1}}$
  - $s \neq S_0$
  - $\{(\neg)F_1(s), (\neg)F_2(s), \ldots, (\neg)F_m(s), (\neg)(f_1(s) = y_1), (\neg)(f_2(s) = y_1), \ldots,$
    $(\neg)(f_m(s) = y_m), \{\Phi_i\}_{i \in \{1..l\}}\} \vdash \neg\Phi(s)$
- *Argument:*
  $\{\{A_{ss}^{F_i}\}_{i \in \{1..n\}}, \{(\neg)\Phi_{F_i}(s')\}_{i \in \{1..n\}}, \{A_{ss}^{f_i}\}_{i \in \{1..m\}}, \{(\neg)(\Phi f_i(ss) = y_i)\}_{i \in \{1..m\}},$
  $\{\Phi_i\}_{i \in \{1..l\}}, s = do(a, s')\} \vdash \neg\Phi(s)$

The notation $(\neg)$ means that the symbol $\neg$ may or may not be present. $A_{ss}^{F_i}/A_{ss}^{f_i}$ denotes the relevant to the relational/functional fluent F/f Successor State axiom, and $\{\Psi_i\}_{i \in \{1..n\}}$ denotes a sequence of n statements of the form Ψ.

Domain Level Dispute. Ultimately, PLDs identify disagreements about the domain. These disagreements may involve differences in the operator specification, conflicting initial situation beliefs, or contradicting situation-less statements. To resolve such disagreements, agents proceed to the domain level phase of the dialogue.

Statement does Not Hold Initially arguments explain why a statement regarding initial situation beliefs is incorrect. The justification is based on formulas regarding initial state beliefs of relevant situation-less beliefs.

Statement does Not Hold Initially (NHI)

- *Preconditions:* $\Phi(S_0) \in H_{v_{k-1}}$
- *Argument:* $\{\Phi'_1(S_0), \ldots, \Phi'_n(S_0)\} \vdash \neg\Phi(S_0)$

Situation-less Statement does Not Hold arguments explain why a statement regarding non-fluent facts is not correct. The justification is based on other

timeless formulae.

Situation-less Statement does Not Hold (NH)

- *Preconditions:* $\Phi \in H_{v_{k-1}}$
- *Argument:* $\{\Phi'_1, \Phi'_2, \ldots, \Phi'_n\} \vdash \neg\Phi$

Invalid Successor State axiom arguments can be used to attack statements regarding the effects of actions. They explain that a proposed successor state axiom is wrong as it does not match the correct axiom.

Invalid Successor State axiom (ISS)

- *Preconditions:*
  - $F(x_1, x_2, \ldots, x_n, do(a, s)) \equiv \Phi_F(x_1, x_2, \ldots, x_n, a, s) \in H_{v_{k-1}}$ or
  - $F(x_1, x_2, \ldots, x_n, do(a, s)) \equiv \Phi_F(x_1, x_2, \ldots, x_n, a, s)$
  - $\neg(\Phi'_F(x_1, x_2, \ldots, x_n, a, s) \equiv \Phi_F(x_1, x_2, \ldots, x_n, a, s))$ or
  - $\neg(\Phi'_f(x_1, x_2, \ldots, x_n, y, a, s) \equiv \Phi_f(x_1, x_2, \ldots, x_n, y, a, s))$
- *Argument for relational fluents:*
  $\{F(x_1, x_2, \ldots, x_n, do(a, s)) \equiv \Phi'_F(x_1, x_2, \ldots, x_n, a, s)\} \vdash$
  $\neg(F(x_1, x_2, \ldots, x_n, do(a, s)) \equiv \Phi_F(x_1, x_2, \ldots, x_n, a, s))$
- *Argument for functional fluents:*
  $\{\neg f(x_1, x_2, \ldots, x_n, do(a, s)) = y \equiv \Phi_f(x_1, x_2, \ldots, x_n, y, a, s)\} \vdash$
  $\neg(f(x_1, x_2, \ldots, x_n, do(a, s)) = y \equiv \Phi_f(x_1, x_2, \ldots, x_n, a, s))$

Invalid Action Precondition axiom arguments can attack statements regarding the preconditions of planning operators.

Invalid Action Precondition axiom (IAP)

- *Preconditions:*
  - $Poss(A, s) \equiv \Pi_A(x_1, x_2 \ldots, x_n, s) \in H_{v_{k-1}}$
  - $\neg(\Pi'_A(x_1, x_2 \ldots, x_n, s) \equiv \Pi_A(x_1, x_2 \ldots, x_n, s))$
- *Argument:*
  $\{Poss(A, s) \equiv \Pi'_A(x_1, x_2 \ldots, x_n, s)\} \vdash \neg(Poss(A, s) \equiv \Pi_A(x_1, x_2 \ldots, x_n, s))$

If the opponent cannot make any valid DLD attacks, and cannot backup to attempt an alternative attack, then the dispute is a failing attack. Therefore, the plan proposal that initiated the dispute as well as the arguments that the proponent employed to support it are acceptable.

4 Parcel World Domain Example

In this section, we present a simple example, revealing the key aspects of how the framework works. Two agents need to decide on a plan that will get an object processed. Agent A is the delivery agent, and is able to move($\Uparrow$, $\Downarrow$, $\Rightarrow$, $\Leftarrow$), *pickup* and *deliver* parcels. Agent B is the processing agent and can only *process* deliveries. Actions *pickup* and *deliver* have no preconditions, but produce conditional effects depending on the position of the object. The *process* action requires B to hold the object. Agents have conflicting beliefs about the position

of the object. The shared objective of the two agents is to process the object. The initial state of the world for agents A and B is described in Figure 4.

Agent A constructs a plan and makes a proposal. B validates the plan and discovers that the final action of the plan is inapplicable. Figure 4 describes the state of the world after each action in the plan for the two agents. Figure 4

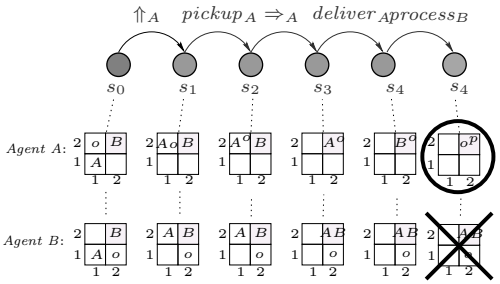

Fig. 2. How a plan affects the environment according to each agent's beliefs in the Parcel World Domain example

describe the argumentation-based dialogue regarding the validity of the plan and the beliefs supporting it. If A convinces B, the plan will be followed; otherwise, replanning will be required. The dialogue shows that B will attack the proposal, because she believes that she will not have the object in the situation that the process action will be performed. A disagrees and bases her next argument on statements about the previous situation, explaining that B will have the object, since it will be delivered to her. The messages show that each *NHS* argument is supported by statements regarding a situation closer to the initial situation. Eventually, an initial situation belief disagreement is identified. Resolution can be attempted through belief argumentation.

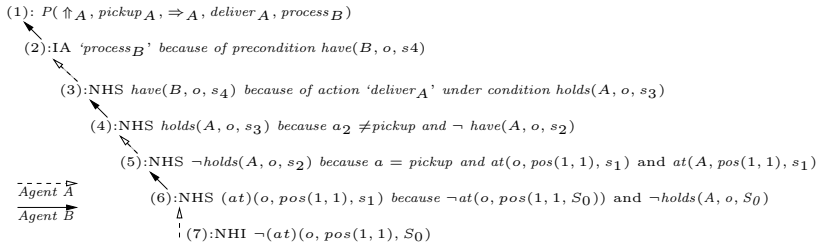

Fig. 3. Messages exchanged by the agents in the Parcel World Domain example

5 Protocol properties

In this section, we will present some properties of the dialogue. In order to do so, the introduction of assumptions is necessary.

5.1 Assumptions

We consider the beliefs of each agent to be consistent. So there cannot be contradictory statements following from an agent's beliefs. We consider cooperative agents, which are truthful with respect to the facts that they introduce to the dialogue. They can employ statements that follow from their beliefs or appear in the other party's commitments. We also assume that the agents are trying to achieve a common goal, i.e. all plans that are discussed have to achieve this predefined goal.[4]

We consider the agents to have a confident assertion attitude and a sceptical acceptance attitude. Agents with a confident assertion attitude will present any valid argument they can construct. Sceptical acceptance attitude ensures that only claims of acceptable arguments will be accepted. Agent attitudes are necessary because protocols usually define preconditions that make some argument moves available to an agent, but do not prescribe which of the available moves an agent will actually perform. By assuming underlying attitudes we can predict what these moves will be. The effects of different agent attitudes on dialogues have been studied in [13].

We follow the Causal Completeness Assumption, which ensures that all the effects of the planning operators are explicitly enumerated in the Successor State axioms. This assumption is employed to treat the *ramification problem* in the same way treated in [10]. Therefore, there are no formulae that can be inferred from the effects of any action and are not listed explicitly through the Successor State axioms. In other words, there is no distinction between direct and indirect action effects (if this was the case, a standard planner would not be able to construct a plan in every case, since only direct effects would be considered during planning; see [8] for planning under defeasible knowledge).

5.2 Properties

The following properties explain that the proposed protocol produces sound results. Completeness has to do with the construction of the proposals, and is closely related to the planning problem. In this paper we do not focus on planning. We consider agents that construct proposals through conventional planning methods [14]. The protocol enables the exchange of proposals for any plans that can be expressed in Situation Calculus and follow the relevant definitions.

Proposition 1. *If a plan is acceptable to both agents, no argument attacking the proponent's proposal argument will be raised by the opponent.*

[4] Note that other protocols could be used for goal negotiation prior to the use of the protocols we suggest, if necessary.

Proof. From the specification of the arguments and the dialogue moves, the opponent can attack a 'Proposal' argument (P) either by raising an 'Invalid Action' (IA) or a 'Not holds in Situation s' (NHS) argument. An attack can be made if OPP believes that either an action a proposed by the proponent is not applicable, $\mathcal{D}_{OPP} \models \neg Poss(a, s)$, or the plan does not achieve the goal, $\mathcal{D}_{OPP} \models \neg G(s_n)$. This contradicts the claim that the plan is acceptable to the opponent, according to Definition 3.

The following proposition identifies bounds on the length of Plan Level Dispute (PLD) sub-dialogues, showing that it is linear in the number of actions in the plan.

Proposition 2. *If the plan is not acceptable, a finite number of PLD arguments will be exchanged. The length of the PLD $|M_p|$ will always be $|M_p| \leq n+1$, where n is the number of actions in the plan.*

Proof. The PLD dispute is formed by a sequence of arguments, initiated by P. A PLD sequence is terminated either if a Domain Level Dispute (DLD) argument is raised or if one of the agents is unable to perform a counter move (resulting in either retraction, backup, or dialogue termination). We will show that if the plan is not acceptable with respect to the opponent's beliefs, then a finite sequence of COUNTER moves will be made, until a DLD argument is raised, terminating the PLD. The intuition guiding this proof is that until the dispute focuses on domain beliefs, agents can always perform a valid COUNTER move, which is supported by statements regarding a situation closer to S_0.

From the definition of an acceptable plan, if the plan is not acceptable to the opponent, we can infer that OPP will either believe that one of the proposed actions is not applicable in the relevant situation, $\mathcal{D}_{OPP} \models \neg Poss(a, s)$, or that the plan does not achieve the goal, $\mathcal{D}_{OPP} \models \neg G(s_n)$. In the second case, the disagreement lies in a statement of the form $\Phi(s)$. In the first case, using the relevant Action Precondition axiom, OPP can infer the required support for constructing a IA argument of the form $\{\neg \Pi_a(s), Poss(a(x_1, \ldots, x_n), s) \equiv \Pi_a(x_1, x_2 \ldots, x_n, s)\} \vdash \neg Poss(a, s)$. If the proponent disagrees with the presented Action Precondition Axiom, then an 'Invalid Action Precondition Axiom' (IAP) argument can be raised terminating the PLD. If this is not the case, then the disagreement lies on whether $\Pi_a(s)$ or $\neg \Pi_a(s)$ is true. Again the disagreement lies in a statement of the form $\Phi(s)$.

We will focus on disagreements on statements of the form $\Phi(s)$. In such cases, agent i presented an argument which included the sentence $\Phi(s)$, but the other agent $-i$ disagrees with this statement (i.e. $\mathcal{D}_i \models \Phi(s)$ and $\mathcal{D}_{-i} \models \neg\Phi(s)$). If $s \neq S_0$, and the disagreement is about a non-fluent statement responsible for $\neg\Phi(s)$ (in both cases DLD moves can be constructed terminating the PLD), agent $-i$ may employ the relevant Successor State axiom to show that some relational or functional fluents ($F(s)$ or $f(s)$) take particular values in s that make $\Phi(s)$ false. To construct the necessary support, $-i$ identifies the conditions, which are specified in each relevant Successor State axiom (either $\Phi_F(s')$ or $\Phi_f(s')$), which hold in the previous situation s'.

Since the agents disagree about $\Phi(s)$, they would disagree also on some statements regarding the conditions in the previous sate or on whether the employed axioms are correct. If the second is the case, agent i would introduce an 'Invalid Successor State axiom' argument, initiating a DLD. In the first case, disagreement lies again on a statement $\Phi(s')$.

Therefore, if a DLD argument is not introduced, the agent that needs to make the next move can always come up with a *NHS* argument attacking a statement the previous argument put forward by the other party. The statements mentioned in the support of this argument (and can be attacked by a PLD argument) refer to situation s', the fluents that are in the claim of the argument refer to s, and $s = do(a, s')$. Each attack made by a *NHS* argument is justified by statements regarding a situation closer to the initial situation. Therefore, since *NHS* arguments cannot attack statements regarding the initial situation, the maximum length of a sequence of consecutive *NHS* statements is n, if the argument initiating the sequence attacks a statement referring to situation s_n. Moreover, from the specification of the argument types, 'Proposal' and the 'Invalid Action' arguments can only appear once in a sequence of COUNTER moves. Therefore, the amount of consecutive PLD arguments can be at most $n + 1$. Such a sequence, will always result in a DLD argument, since agents are always able to construct either a PLD or a DLD argument when presented with a PLD argument. All arguments in a PLD argument sequence will be unique, since they refer to statements regarding different situations.

Proposition 3. *The proponent will present a successful (credulous) defence of the plan proposal if and only if either (i) the plan is acceptable with respect to the opponent's beliefs, or (ii) if the dispute terminates as a failing attack of the plan proposal argument.*

Proof. If the plan is acceptable, then no argument will be raised against the plan proposal argument (Proposition 1). *PRO* wins the dispute.

If the dispute terminates as a failing attack of the plan proposal argument, then the final argument is presented by the proponent. Therefore, the length of the dialogue is odd, and *PRO* wins the dispute.

If the plan is not acceptable with respect to the opponent's beliefs, and the dispute terminates as a failing defence of the plan proposal argument, then the opponent would have presented an argument that the proponent could not counter, or backup. This will be the final move of the dispute, making the number of moves even. Therefore, the opponent would win the dispute.

Proposition 4. *If the dispute for a Proposal argument terminates as a failing attack after move k, then the plan will be acceptable with respect to the set of beliefs* $D_{acc} = D_{OPP} \backslash \{\Phi(S_0) | (\exists \langle H, h \rangle \in P_k).[(\exists \Phi'(S_0) \in H \cup \{h\}).\Phi(S_0) \wedge \Phi'(S_0) \vdash \bot]\} \cup \{\Phi(S_0) | (\exists \langle H, h \rangle \in P_k).[(\exists \Phi'(S_0) \in H \cup \{h\})]\}$

Proof. Since all runs are failing attacks, P_k is the set of the acceptable arguments the proponent presented during the dialogue. In addition, a Domain Level Dispute phase would have been initiated for every domain belief that is not shared

among the agents and is relevant to the plan. Let D^{conf}_{OPP} be the set of all the statements from the domain beliefs of the opponent that conflict with statements appearing in an argument in P_k. These are the beliefs that were responsible for OPP not accepting the plan initially.

The plan will be acceptable with the set containing all statements appearing in P_k and all statements in D_{OPP} that do not conflict with the statements from P_k. By excluding all statements that were used to construct the arguments attacking the plan, and the arguments that supported these, we have excluded all the statements that are responsible for the arguments that were employed by the opponent. Therefore, no other arguments attacking the plan can be constructed using the opponent's remaining beliefs, since if this was the case, the opponent would have constructed them in move k continuing the dialogue (since we consider agents with confident assertion attitude).

6 Argument Generation

In order for the protocol to be effective, the agents need efficient mechanisms for argument generation. In general, argument generation is a complex task. Even with the assumptions that i)the agents' knowledge bases are propositional and ii) conflict-free, argument generation is co-NP-complete, since it is as complex as finding a proof in propositional logic (a further analysis can be found in [13]). Moreover, the support of such arguments will chain back to beliefs about the initial situation, resulting in extensive proofs, if there are no heuristics to restrict them. In this section, we will provide an algorithm for argument generation related to plans and will explain how this algorithm guides the search through the agents beliefs. The algorithm is based on the evaluation of the executability of all the actions in the plan, and on the projection of the actions' effects, in order to evaluate the values of statements in the resulting situations. The complexity of these processes is related to the expressive power of the employed planning formalism [15].

After being presented with an argument, an agent searches for valid attackers. The task of argument generation can be performed by searching through the plan projection data. The search the agent has to conduct can be guided, depending on the type of the received argument. The constructed argument must follow the specification of the employed dialogue move (as for example the rules regarding repetition of arguments).

If the agent receives a plan proposal, it can attack it if she identifies that according to her beliefs an action is not possible to be performed, using a *IA* argument, or if she believes that the plan does not succeed in achieving the goal. Differences of opinion regarding the operator specification (i.e. Successor State axioms or Action Precondition axioms) can be identified through the introduction of *ISS* or *IAP* arguments. Arguments supported by statements of the form $\Phi(s)$ that the agent believes to be incorrect (the validity of which may be evaluated through projection), can be attacked by *NH*, *NHI* or *NHS* arguments.

switch *Type of received argument* v_{k-1} **do**
case P
  if $Poss(a, s) \in H_{v_{k-1}}$ *and* $\mathcal{D}_i \models \neg Poss(a, s)$ **then return** IA for a and relevant A_{ap};
  if $\mathcal{D}_i \models \neg G(S_n)$ **then return** NH or NHS argument for the relevant fluent(s) and A_{ss} that explain $\neg G(s_n)$;
case IA
  if $A_{ap} \in H_{v_{k-1}}$ *and* $A_{ap} \notin \mathcal{D}_i$ **then return** IAP argument for correct A_{ap} ;
  if $\Phi(s) \in H_{v_{k-1}}$ *and* $\mathcal{D}_i \models \neg\Phi(s)$ **then return** NH, NHI or NHS argument for the relevant fluent(s) and A_{ss} that explain $\neg\Phi(s)$;
case NHS
  if $A_{ss} \in H_{v_{k-1}}$ *and* $A_{ss} \notin \mathcal{D}_i$ **then return** ISS argument for correct A_{ss} ;
  if $\Phi(s) \in H_{v_{k-1}}$ *and* $\mathcal{D}_i \models \neg\Phi(s)$ **then return** NH, NHI or NHS argument for the relevant fluent(s) and A_{ss} that explain $\neg\Phi(s)$;
case NHI **if** $\Phi(S_0) \in H_{v_{k-1}}$ *and* $\mathcal{D}_i \models \neg\Phi(S_0)$ **then return** NH/NHI argument for $\neg\Phi(S_0)$;
case NH **if** $\Phi \in H_{v_{k-1}}$ *and* $\mathcal{D}_i \models \neg\Phi$ **then return** NH argument against Φ;
case ISS **return** ISS argument for correct A_{ss};
case IAP **return** IAP argument for correct A_{ap};
return no move;

Algorithm 1: Argument generation algorithm

In order to construct such arguments, the agent needs to identify the symbols that appear in $\Phi(s)$ and are responsible for $\Phi(s)$ to be false. If s is not S_0, then the agent needs to construct a proof, explaining that the necessary conditions hold in the previous situation, so that these symbols have certain values in the resulting situation that falsify $\Phi(s)$. In this case a *NHS* argument can be constructed. If s is S_0, then the statement is about the initial situation and the agent can respond with a *NHI* argument. Finally, if the statement is false because of non-fluent statements, a *NH* argument can be employed.

Argument generation for the DLD subdialogues will need to be based on logical proofs, since it involves unstructured initial world beliefs. In this case, a large subset of the agent's beliefs are no longer relevant, since for DLD argument generation, only beliefs regarding the initial state of the world should be considered. All beliefs regarding the operator specification and the planning theory are irrelevant. This greatly reduces the set of the statements the agents have to search over when constructing a DLD argument. Moreover, if argument generation is implemented in propositional logic, the first order representation for all statements needs to be translated to propositional logic. The size of a propositional representation is exponentially larger in the worst case than the size of a corresponding first-order representation. This means that the propositional representation is considerably smaller without the operator specification and the planning theory axioms, i.e. a naive inclusion of planning axioms in propositional logic would be completely intractable for any real-world domain.

7 Conclusion

In this paper, we presented a novel dialogue protocol that allows cooperating agents to jointly evaluate plan proposals and to identify relevant conflicts in agents' beliefs. Moreover, we proposed efficient search-based algorithms for argument generation based on the specific characteristics of the planning domain.

Our approach is influenced by recent work on argumentation for practical reasoning and deliberation [2–4] and planning over defeasible knowledge [8]. While related to this work, we maintain a closer relation to classical AI planning, as no assumptions are made regarding plan generation and therefore any efficient planner can be employed for this task.

In the future we would like to investigate different strategies for argument selection, in cases where the opponent has a choice over various *IA* or *NHS* moves. In addition, our research can be complemented with heuristics for planning under uncertainty, considering multiagent planning among cooperative agents with different beliefs. It would be interesting to extend our protocol to include moves that enable distributed planning, especially in cases in which there is no agent that can construct a plan alone.

References

1. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77**(2) (1995) 321–357
2. Atkinson, K., Bench-Capon, T.: Practical reasoning as presumptive argumentation using action based alternating transition systems. AI **171**(10-15) (2007) 855–874
3. Rahwan, I., Amgoud, L.: An argumentation based approach for practical reasoning. In: Proceedings of AAMAS-06, New York, NY, USA, ACM (2006) 347–354
4. Tang, Y., Parsons, S.: Argumentation-based dialogues for deliberation. In: Proceedings of AAMAS-05, USA, ACM (2005) 552–559
5. Pynadath, D.V., Tambe, M.: The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. Journal of Artificial Intelligence Research **16** (2002) 389–423
6. Durfee, E.: Distributed Problem Solving and Planning. In Weiß, G., ed.: Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. MIT Press, USA (1999) 121–164
7. Lesser, V., Decker, K., Carver, N., Garvey, A., Neiman, D., Prasad, M., Wagner, T.: Evolution of the GPGP domain-independent coordination framework. Technical Report 98-05, Computer Science Department, UMASS (1998)
8. Garcia, D., Garcia, A., Simari, G.: Planning and defeasible reasoning. In: Proceedings of AAMAS-07, USA, ACM (2007)
9. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence **4** (1969)
10. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press (2001)
11. Dunne, P.E., Bench-Capon, T.: Two party immediate response disputes: properties and efficiency. Artificial Intelligence **149**(2) (2003) 221–250
12. Vreeswijk, G., Prakken, H.: Credulous and sceptical argument games for preferred semantics. In: Proceedings of JELIA '00, UK, Springer-Verlag (2000) 239–253
13. Parsons, S., Wooldridge, M.J., Amgoud, L.: An analysis of formal inter-agent dialogues. In: Proceedings of AAMAS-02, USA, ACM (2002) 394–401
14. Nau, D., Ghallab, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
15. Nebel, B.: On the compilability and expressive power of propositional planning formalisms. Journal of Artificial Intelligence Research **12** (2000) 271–315