Dialectical Theory for Multi-Agent Assumption-based Planning

Damien Pellier, Humbert Fiorino

Laboratoire Leibniz, 46 avenue Félix Viallet F-38000 Grenboble, France
{Damien.Pellier,Humbert.Fiorino}.imag.fr

Abstract. The purpose of this paper is to introduce a dialectical theory for plan synthesis based on a multi-agent approach. This approach is a promising way to devise systems based on agent planners in which the production of a global shared plan is obtained by conjecture/refutation cycles. Contrary to classical approaches, our contribution relies on agents' dialectical reasoning: in order to take into account the partial knowledge and the heterogeneous skills of the agents, we propose to consider the planning problem as a defeasible reasoning where agents exchange proposals and counter-proposals and are able to conjecture *i.e.*, formulate plan steps based on hypothetical states of the world. The dialogue between agents is a joint investigation process allowing agents to progressively prune objections, solve conjectures and elaborate solutions step by step.

1 Introduction

The problem of plan synthesis achieved by autonomous agents in order to solve complex and collaborative tasks is still an open challenge. Increasingly new application areas can benefit from this research domain: for instance, cooperative robotics [1] or composition of semantic web services [2] when considering actions as services and plans as composition schemes. From our point of view, multi-agent planning can be likened to the process used in automatic theorem proving. In a sense, a plan can be considered to be a particular proof based on specific rules, called actions. In this paper, we draw our inspiration from the proof theory described by Lakatos. According to [3], a correct proof does not exist in the absolute. At any time, an experimentation or a test can refute a proof. If one single test leads to a refutation, the proof is reviewed and it is considered as mere conjecture which must be repaired in order to reject this refutation and consequently to become less questionable. The new proof can be subsequently tested and refuted anew. Therefore, the proof elaboration is an iterative non monotonous process of conjectures - refutations - repairs.

The same is true of our approach. The plan synthesis problem is viewed as a dialectical and collaborative goal directed reasoning about actions. Each agent can refine, refute or repair the ongoing team plan. If the repair of a previously refuted plan succeeds, it becomes more robust but it can still be refuted later. If the repair of the refuted plan fails, agents leave this part of the reasoning and explore another possibility: finally "bad" sub-plans are ruled out because there is no agent able to push the investigation process further. As in an argumentation with opponents and proponents, the current plan

is considered as an acceptable solution when the proposal/counter-proposal cycles end and there is no more objection.

The originality of this approach relies on the agent's capabilities to elaborate plans under partial knowledge and/or to produce plans that partially contradict its knowledge. In other words, in order to reach a goal, such an agent is able to provide a plan *which could be executed if certain conditions were met*. Unlike "classical" planners, the planning process does not fail if some conditions are not asserted in the knowledge base, but rather proposes an Assumption-Based Plan or *conjecture*. Obviously, this conjecture must be *reasonable*: the goal cannot be considered "achieved" and the assumptions must be as few as possible because they become new goals for the other agents. For instance, suppose that a door is locked: if the agent seeks to get into the room behind the door and the key is not in the lock, the planning procedure fails even though the agent is able to fulfill 100% of its objectives behind the door. Another possibility is to suppose for the moment that the key is available and then plan how to open the door etc. whereas finding the key might become a new goal to be delegated. To that end, we designed a planner that relaxes some restrictions regarding the applicability of planning operators.

Our approach differs from former ones in two points. First of all, unlike approaches that emphasize the problem of controlling and coordinating a posteriori local plans of independent agents by using negotiation [4], argumentation [5], or synchronization [6] *etc.*, the dialectical theory for plan synthesis presented here focuses on generic mechanisms allowing agents to jointly elaborate a global shared plan and carry out collective actions. Secondly, by elaboration, we mean plan production and not instantiation of predefined global plan skeletons [7, 8]. This is achieved by composing agents' skills *i.e.*, the actions they can execute for the benefit of the group. Thus, the issues are: how can agents produce plans as parts of the global proof with their partial and incomplete beliefs? what kind of refutations and repairs agents can propose to produce robust plans? and how to define the conjecture - refutation protocol so as to converge to an acceptable solution plan?

In this paper, we introduce a multi-agent assumption-based planning approach. In section 2, we present the primary notions used in this approach. Then, in section 3, we define the concept of *proof board* used by agents to collaboratively build a solution plan and finally, in section 4, the dialectical mechanisms for the conjecture-refutation process is presented.

2 Primary Notions

We start by defining the language used to describe agents' beliefs. This language is based on a first-order language $\mathcal{L}$ in which there is a finite number of predicates symbols and constants symbols but no function symbols. A *state* is a set of ground atoms of $\mathcal{L}$. Since $\mathcal{L}$ has no functions symbols, the set S of all possibles states s is guaranteed to be finite. An atom p holds in s iff $p \in s$. If g is a set of literals (*i.e.*, atoms and negated atoms), we will say that s *satisfied* g (denoted $s \models g$).

Now, let us introduce, the definition of a planning operator used by agents. An planning operator defines a transition operation from a state to another one.

Definition 1 (Planning Operator). *A* planning operator *is a triple* $o = \langle$*name*(o), *precond*(o), *effects*$(o)\rangle$ *whose elements are as follows:*

- *name*(o)*, the name of the operator,* $n(x_1, \ldots, x_k)$ *where* n *is a symbol and* $x_1, \ldots, x_k$ *define operator's parameters.*
- *precond*(o) *and effects*(o)*, the* preconditions *and* effects *of* o*, respectively defining the literals that must be held in the state where the operator is applied and the literals that must be added (denoted effect*$(o)^+$*) or removed (denoted effect*$(o)^-$*), to compute the transition operation.*

Although we use the same operator representation as in classical planning, the operator semantic in our approach is different. In classical planning, an operator is applicable to a state s if o is ground and s is a state such precond$(o) \subseteq s$. Our approach relaxes this constraint: all operators are applicable to a state s. Hence, we must distinguish facts that hold in s and facts that do not hold. The second are called *assumptions*. An assumption defines a literal $p \in$ precond(o) such p do not hold in s. We use assump(o) to denote the set of assumptions needed to apply an operator o in a particular state s. The state resulting of the application of o to s_i is the state:

$$s_{i+1} = ((s_i \cup \text{assump}(a)) - \text{effects}^-(a)) \ \cup \ \text{effects}^+(a)$$

For instance, consider the initial belief state of an agent $s_0 = \{$at(cont,loc1)$\}$ and a simple operator that can be performed by this agent to move a container from a location to another one: name(o) = move(*c*,*l1*,*l2*); precond(o) = $\{$connected(*l1*,*l2*), at(*c*,*l1*)$\}$ and effect(o) = $\{\neg$at(*c*,*l1*), at(*c*,*l2*)$\}$. In this example, the agent has no information about the connection between the locations loc1 and loc2. In order to apply the move operator, the agent must assume the assumption connected(loc1,loc2). The state resulting of the application of the move operator is the state: $s_1 = \{$connected(loc1, loc2), at(cont,loc2)$\}$.

Before going further and introducing our multi-agent planning model, we must clarify one point. We say that an assumption is a precondition of an operator o that do not hold in the state s where the operator is applied. Thus, there are two cases: i) if a precondition p is not contained in s, the fact must be added to the agent's belief and simply considered as a *hypothetical fact*; ii) if a precondition does not hold because its negation is contained in s, the agent must first remove the negation before adding the precondition. We call this kind of assumption a *fact negation*.

Assumptions are important opportunities for improving collaborative synergy between agents. They can be refined by the other agents in order to produce the supposed facts (*e.g.*, by connecting the two locations loc1 and loc2). They are viewed as subgoals that must be fulfilled by other agents.

Definition 2 (Agent). *An* agent *is a triple* $ag = \langle$*name*(ag), *operators*(ag), *beliefs*$(ag)\rangle$, *where:*

- *name*(ag)*,the name of the agent;*
- *operators*(ag)*, a set of operators, i.e., the skills of the agent;*
- *beliefs*(ag)*, a set of literals, i.e., the initial beliefs of the agent.*

In classical planning, a planning domain is defined by a set of operators. In our approach, operators are included in agents' description. Thus, we define a multi-agent planning domain as a set of agents.

Definition 3 (Multi-Agent Planning Domain). *A* multi-agent planning domain $\mathcal{D}$ *is defined as a set of agents.*

Finally, we need to define the notion of multi-agent planning problem. A multi-agent planning problem must define the goals that must be reached and the set of agents that must solve it.

Definition 4 (Multi-Agent Planning Problem). *A* multi-agent planning problem *is a couple* $\mathcal{P} = \langle \mathcal{AG}, g \rangle$*, where:*

- $\mathcal{AG}$ *defines a set of agents' names;*
- g *is a set of literals that must be reached by the agents defined in* $\mathcal{AG}$.

Consider a simple domain containing four agents: a farmer, a miller, a baker and a conveyor. The farmer sows wheat, which must be harvested. The miller grinds the farmer's wheat to produce flour. The baker makes bread with miller's flour and finally the conveyor is in charge of moving the goods needed by the other agents. An instance of a multi-agent planning problem can define with $\mathcal{AG} = \{\text{famer}, \text{miller}, \text{baker}, \text{convoyor}\}$ and $g = \{\text{has-goods}(\text{baker}, \text{bread}, 2)\}$.

3 Conjectures Space Search

The plan synthesis relies on dialectical exchanges between agents as expected in a debate. Agents interact collaboratively in the dialogue so as to construct a plan without assumption, fulfilling the assigned goals. In order to build such a plan and organize the dialog between agents, we need a structure, called *proof board*. This structure has two main functions: it must be able to represent the space search as in classical planning and it must be able to specify the dialectical rules used by agents to interact.

3.1 Conjectures and Plans

First, let us refine the notion of conjecture used in our approach. We have informally introduced a conjecture as a plan that can be executed if certain conditions were met. In classical planning, a plan is a set of ground operators organized into some structure, *e.g.,* a sequence. However, a sequence of operators is a particular plan that reflects the intrinsic constraints of the operators. It seems to be to much restrictive for a multi-agent approach of collaborative planning, *e.g.,* it is no possible to define concurrent actions. Therefore, to find out what is needed in a conjecture, consider an informal planning step (shown figure 1) on the simple example previously introduced with the farmer, the miller, the baker and the conveyor.

baker_1 :	"I can make 2 breads to solve the goal, but I need 2 flour containers available in loc1."
conveyor_1 :	"I can transport the flours containers at loc1, but I don't know where I must load the goods."
miller_1 :	"I propose to sell you the flours containers. I needed to be payed 4 euros for that and find someone to transport flour containers from loc2 to loc1. Moreover, I need a wheat container available in loc2 to grind the flour."

baker$_2$: “Thank you for your help, miller, but I have not enough money.”
miller$_2$: “Ok, give me only 2 euros.”
baker$_2$: “Good deal, I pay you.”
conveyor$_2$: “Thus, I understand that I must load the flour in loc2.”
farmer$_1$: “I propose to sell you a wheat container. I need to be payed 1 euros for that and find someone to transport the container from loc3 to loc2.”
miller$_3$: . . .

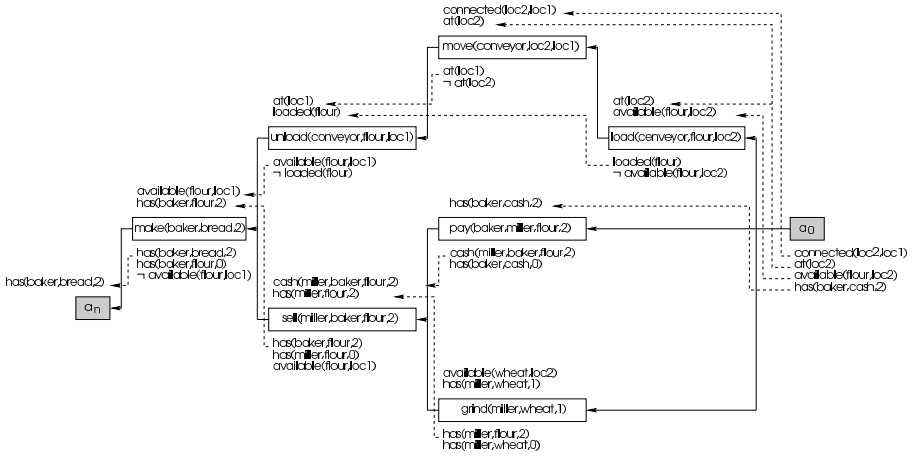

Fig. 1. Example of conjecture: each boxes is an operator with preconditions above and effects below. Solid arrows are ordering constraints, dashed arrows are causal links and binding constraints are implicit or shown directly in the operator parameters. This representation is based on [9].

Operators. Initially, baker$_1$ proposes to add the operator make-bread to reach the goal g = { has(baker,bread,2)}. This operator make two assumptions: available(flour, loc1) and has(baker,flour,2). These assumptions must be refined. Thus, conveyor$_1$ and miller$_1$ propose recursively to add others operators or sub-conjecture to reach these two new goals.

Ordering Constraints. Consider the sub-conjecture added by conveyor$_1$; it achieves its purpose only if it is constrained to come *before* the make-bread operator. But should this conjecture come before or after the miller conjecture? Both options are possible. We use the least commitment principle of not adding constraints unless it is strictly needed. If no constraint are specified the conjecture between conveyor$_1$ and miller$_1$, these two conjectures will be able to run concurrently.

Causal links. Because there is no explicit notion of current state (distributed on the agents), an ordering constraint does not say, for instance, that the flour stays available at loc1 until make-bread operator is performed. Hence, we need to encode explicitly in the conjecture the reason why the conveyor$_1$ sub-conjecture was added: to satisfy the assumption available(flour, loc1). The relation between the baker’s conjecture and the conveyor’s one with respect to available(flour, loc1), is called a *causal link*.

Binding Constraints. Operators are added in a conjecture with systematic variable renaming. For instance, we must ensure that the conveyor conjecture concerns the same container flour and the same location loc1 as those in operator make-bread.

Definition 5 (Conjecture). *A* conjecture *is a tuple $\chi = \langle \mathcal{A}, \prec, \mathcal{B}, \mathcal{C} \rangle$, where:*
- *$\mathcal{A} = \{a_1, \ldots, a_k\}$ is a set of partially instantiated operators.*
- *$\prec$ is a set of ordering constraints on $\mathcal{A}$ of the form ($a_i \prec a_j$).*
- *$\mathcal{B}$ is a set of binding constraints on $\mathcal{A}$ of the form $x = y$, $x \neq y$ or $x \in D_x$, where D_x is the domain of x.*
- *$\mathcal{C}$ is a set of causal links of the form ($a_i \xrightarrow{p} a_j$), such that a_i and a_j are operators in $\mathcal{A}$, the constraint $a_i \prec a_j$ is in $\prec$, assumption p is an effect of a_i and a precondition of a_j and finally the binding constraints between a_i and a_j about p are in $\mathcal{B}$.*

The proof board is a conjecture space defining a directed graph whose vertices are conjectures and whose edges correspond to the *transition operation* proposed by the agent. An outgoing edge from a vertex χ is a transition operation that transforms χ into a successor χ'. A transition operation can be: a refinement (*i.e.,* adding operators to prove an assumption), a refutation (*i.e.,* highlighting inconsistencies in the conjecture) and a repair of a previously highlighted inconsistency. Therefore, multi-agent assumption-based planning is a search in the proof board from a initial conjecture to a node recognized as a solution plan. Note that due to no explicit current state representation, goals and initial state must be defined as particular conjectures. Since preconditions are possibly assumptions, the propositions corresponding to the goals are represented as preconditions of a dummy operator a_n. Similarly, the initial state is represented as the effects of a dummy action a_0. The effects of a_0 define the union of the agents' beliefs. We make the assumption that the agents' beliefs are consistent.

3.2 Solution Plan

Let us now specify what is a solution plan to a planning problem $\mathcal{P} = \langle \mathcal{AG}, g \rangle$. A solution plan is a conjecture that has particular properties. First, a conjecture is a solution plan if the conjecture makes no assumption. But according to the conjecture definition, it is not enough. A solution conjecture must define a consistent set of ordering constraints, binding constraints and causal links. These properties allow us to define three kinds of refutations.

Proposition 1 (Solution Plan[1]). *A conjecture $\chi = \langle \mathcal{A}, \prec, \mathcal{B}, \mathcal{C} \rangle$ is a solution plan to a planning problem $\mathcal{P} = \langle \mathcal{AG}, g \rangle$, if χ has no assumption and χ can not be refuted.*

Definition 6 (Ordering Refutation). *An action a_k of a conjecture χ refutes an ordering constraint $a_i \prec a_j$ iff $a_k \prec a_i$ and $a_j \prec a_k$.*

Definition 7 (Binding Refutation). *An action a_k of a conjecture χ refutes an binding constraint iff one of the following condition holds:*
1. *if there is an operator a_k that contains a variable x such that $x \in D_x$ and x is not consistent with $\mathcal{B}$.*

[1] can be proved inductively on the number of operators in $\mathcal{A}$.

2. *if there is an operator a_k that contains two variables x and y such that $x = y$ is not consistent with $\mathcal{B}$.*
3. *if there is an operator a_k that contains two variables x and y such that such that $x \neq y$ is not consistent with $\mathcal{B}$.*

Definition 8 (Causal Refutation). *An action a_k of a conjecture χ refutes a causal link $a_i \xrightarrow{p} a_j$, iff:*

- *a_k has an effect $\neg q$ and $\neg q$ is not consistent with p, i.e., p and q are unifiable.*
- *ordering constraints $(a_i \prec a_k)$ and $(a_k \prec a_j)$ are consistent with $\prec$.*
- *binding constraints resulting of the unification of p and q are consistent with $\mathcal{B}$.*

4 Dialectical Mechanisms

In order to tackle the dialectical mechanisms to collaboratively build a solution plan, let us remember the definition of the proof board. The proof board defines a conjectures space where edges represent transition operations: refine, refute or repair. A conjecture is a solution plan if it does not contain assumption and if no agent is able to refute it. This definition gives us two tips to specify the dialectical mechanism. Indeed, the first condition can be reached by refining or repairing. On the contrary, the second condition needs a deliberation process to guarantee that no agent can refute the conjecture. Therefore, we distinguish two layers: i) an *informational layer* that defines the rules to exchange refinements, refutations and repairs about the current conjecture. Each new conjecture suggested by an agent produces new goals to be achieved by the other agents; ii) a *contextualization layer* in which agents can decide to stop interacting when they believe a solution was found or not reachable. Moreover agents can decide to change the dialogue context by forwarding or backtracking into the proof board if the current conjecture has been refuted or none of the agents can refine its assumptions.

4.1 Informational Layer

The characterization of the solution plan brings elements needed for the specification of the speech acts used in the informational layer. The main principle of the multi-agent assumption based planning is to let the agents choose a transition operation to apply to the proof board until χ contains no more assumptions and until χ cannot be refuted. The basic steps of agent's dialectical mechanisms are the following:

- Select a conjecture χ on which to apply a transition operation.
- Select a transition operation to apply to χ.
- Find ways to resolve the transition operation.
- Select a resolver for the transition operation.
- Assert the resolver, *i.e.*, refine, refute or repair.

For each transition operation that can be applied, we introduce a speech act: i) a speech act *refine* is performed by an agent to express the refinement of a conjecture. A refinement can be specified by adding a set of operators, a set of ordering constraints, a set of binding constraints and finally a set of causal links (*e.g.*, miller_1 in example 1); ii) a

speech act *refute* is performed by an agent to express the refutation of a conjecture. A refutation highlights that an action produces a set of ordering inconsistencies or a set of binding inconsistencies or finally a set of causal inconsistencies. The computation of the inconsistencies are based on the formal definition of the three kinds of refutation previously presented (*e.g.*, baker$_2$ in example 1); iii)a speech act *repair* is performed by an agent to express that a conjecture can be repaired by adding and removing respectively a set of operators, a set of ordering constraints, a set of binding constraints and finally a set of causal links (*e.g.*, miller$_2$ in example 1). Note that all informational speech acts can be performed only if they were not already proposed by other agents. This condition guarantees that the proof board defines a loop free directed graph. In order to find ways to resolve a transition operation agents use the following mechanisms:

Refinement. If a conjecture χ contains an operator a_j that makes an assumption p (see figure 2): i) If a causal link ($a_i \xrightarrow{p} a_j$) can be established such that a_i is already in the conjecture, the refinement will contain the causal link ($a_i \xrightarrow{p} a_j$), the ordering constraint ($a_i \prec a_j$) and the binding constraints to unify p with the effects of a_i; ii) Otherwise, agents must compute a sub conjecture χ' to prove p. The refinement will contain all the elements of χ', a causal link ($a_i \xrightarrow{p} a_j$) to specify which operator a_i of χ' reaches the assumption p done by a_j and a ordering constraint ($a_i \prec a_j$). Note that we have already shown in [10] how an agent can produce such conjecture.

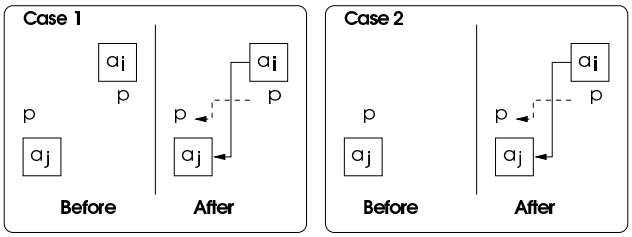

Fig. 2. The left figure shows a refinement when an operator already reached an assumption and right figure shows a refinement by adding a new conjecture.

Repair [2]. If there is a causal refutation on ($a_i \xrightarrow{p} a_j$) by an action a_k that has an effect $\neg q$, and q is unifiable with p, then the resolvers are any of the following: i) add an ordering constraint such that a_k occurs before the causal link; ii) add an ordering constraint such that a_k occurs after the causal link; iii) add a binding constraint that makes q and p non-unifiable.

Refutation. The causal refutation can be computed by testing all triples of actions of a conjecture χ. The ordering refutation can be computed by testing that the ordering constraint represent a loop free graph. Finally, the binding refutation of type 1 and 2 (see definition 7) can be computed in linear time, whereas the type 3 raises a general NP-complete Constraint Satisfaction Problem (CSP).

[2] Repairs of binding refutation and ordering refutation are not discussed here.

4.2 Contextualization Layer

The informational layer defines the basic mechanisms to build a solution plan. Is that enough? Not quite. The dialectical mechanism must guarantee the soundness and the completeness of the collaborative plan synthesis process. Now let us consider the proof board as a search in an AND/OR tree. The assumptions and the refutations correspond to AND branches because all of them must be resolved in order to find a solution. For each assumption and refutation the possible resolvers (*i.e.,* refinement and repair) correspond to OR branches because only one of them is needed in order to find a solution. In order to guarantee the completeness, agents must coordinate their exploration. Therefore, we consider that agents can apply a transition operation only on a specific conjecture in the proof board, called *current conjecture*. This conjecture defines the dialog context. The speech acts define in the contextualization layer allow agents to change the dialog context. We introduce four contextualization speech acts: i) a speech act *prop.solve* is performed by an agent when it believes that a solution plan χ is reached. When the speech act *prop.solve* is proposed each agent checks if it can refute χ. If χ cannot be refuted each agent acknowledges the solve proposition. Otherwise, they refute χ and the dialectical process is extended; ii) a speech act *prop.failure* is performed by an agent when it believes that no solution plan exists. Like the previous speech act, when speech act *prop.failure* is performed, each agent checks if there is a conjecture in the proof board on which they can apply a transition operation. In this case, each agent acknowledges the failure proposition. Otherwise, the dialectical process continues; iii) a speech act *prop.backward* is performed by an agent when it believes that no resolver can be proposed to go further in the current conjecture exploration; iv) a speech act *prop.foreward* is performed by an agent when it believes that agent have no more resolvers to apply at the current conjecture.

Note that all contextualization speech acts define a joint commitment between agents. For instance, all agents must agree on the plan solution before stopping the dialectical plan synthesis process. The computation of the next current conjecture when the speech acts *prop.backward* and *prop.foreward* are proposed by agents is based on A* heuristics. Recall that A* uses a heuristic estimate $f(\chi)$ of the overall solution cost consisting of, in the one hand $g(\chi) =$ cost of the current conjecture χ and in the other hand $h(\chi) =$ estimate of the additional cost of the best complete solution that extends χ. We propose to think $f(\chi)$ as a measure of conjecture *complexity*, *i.e.,* "good" conjecture are simple conjectures. What is significant to compute $f(\chi)$? [11] indicates that the most promising heuristic measure for conjecture selection is the number of actions contained in the conjecture and the number of assumptions done. Therefore, we define $g(\chi)$ as the number of action of χ, *i.e.,* the complexity of the conjecture and $h(\chi)$, the number of assumptions done, since each remaining assumption must be established by some sub-conjecture. Note that this heuristic can be used locally by the agent to choose the best resolver to submit to the other agents.

5 Conclusion

The dialectical plan synthesis theory model presented in this paper relies on plan production and revision by conjecture/refutation cycles: for a given goal, agents try collab-

oratively to produce a valid proof, *i.e.*, a plan. In order to demonstrate the goal assigned to the system, agents interact by using a conventional dialogue approach that can be split in two layers: *informational layer*, which defines the conventions to refine, refute or repair conjectures and *contextualization layer*, which defines the conventions to allow agents to change the dialogue state. The dialogue rules are described according to the *proof board*. The proof board represents the public part of the communication storing the different exchanges between agents. The advantage of the dialectical plan synthesis is to merge in the collaborative plan generation, the composition and the coordination steps. It also includes the notion of uncertainty in the agents' reasoning and allows the agents to make conjectures and to compose their heterogeneous competences. Moreover, we apply conjecture/refutation to structure the multi-agent reasoning as a collaborative investigation process. However, former works on synchronization, coordination and conflict resolution are integrated through the notions of refutation/repairs. From our point of view, this approach is suitable for applications in which agents share a common goal and in which the splitting of the planning and the coordination steps (when agents have independent goals, they locally generate plans and then solve their conflicts) becomes difficult due to the agents strong interdependence.

References

1. Alami, R., Fleury, S., Herrb, M., Ingrand, F., Robert, F.: Multi robot cooperation in the martha project. IEEE Robotics and Automation Magazine **5** (1997) 36–47
2. Wu, D., Parsia, B., Sirin E, Hendler, J., Nau, D.: Automating daml-s web services composition using shop2. In: Proceedings of International Semantic Web Conference. (2003)
3. Lakatos, I.: Proofs and Refutations: The Logic of Mathematical Discovery. Cambridge University Press, Cambridge, England (1976)
4. Zlotkin, G., Rosenschein, J.: Negotiation and conflict resolution in non-cooperative domains. In: Proceedings of the American National Conference on Artificial Intelligence, Boston, Massachusetts (1990) 100–105
5. Tambe, M., Jung, H.: The benefits of arguing in a team. Artificial Intelligence Magazine **20** (1999) 85–92
6. Clement, B., Barrett, A.: Continual coordination through shared activities. In: Proceedings of the International Conference on Autonomous Agent and Muti-Agent Systems. (2003) 57–67
7. Grosz, B., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence **86** (1996) 269–357
8. D'Inverno, M., Luck, M., Georgeff, M., Kinny, D., Wooldridge, M.: The dmars architecture: A specification of the distributed multi-agent reasoning system. Autonomous Agents and Multi-Agent Systems **9** (2004) 5–53
9. Ghallab, M., Nau, D., Traverso, P.: Automated Planning Theory and Practice. Morgan Kaufmann Publishers (2004)
10. Pellier, D., Fiorino, H.: Assumption-based planning. In: In Proceedings of the International Conference on Advances in Intelligence Systems Theory and Applications, Luxemburg (2004)
11. Gerevini, A., Schubert, L.: Accelerating partial-order planners: Some techniques for effective search control and pruning. Journal of Artificial Intelligence Research **5** (1996) 95–137