Multi-Agent Planning with Decommitment*

Gerhard Wickler
AIAI, University of Edinburgh
Edinburgh, Scotland

Antonin Komenda
Agent Technology Center,
Czech Technical University in Prague

Michal Pechoucek
Agent Technology Center,
Czech Technical University in Prague

Austin Tate
AIAI, University of Edinburgh
Edinburgh, Scotland

Jiri Vokrinek
Agent Technology Center,
Czech Technical University in Prague

December 19, 2008

Abstract

One of the problems that must be solved during coalition operations is the planning problem: finding a course of actions for the operation. Knowledge-based planning is a technique that can be used to address this problem. However, traditional planning has focussed on the finding of a plan that achieves a given goal or accomplishes a given task. During a coalition operation such a plan may need to be agreed between the different parties involved in the coalition. In this paper we shall describe an approach that can be used for finding an *agreed* plan. This plan will include a set of decommitment rules allowing participants to specify conditions under which they are allowed to deviate from the agreed course of action. The proposed algorithm can be used as the basis for an automated planner, but it should also be usable in a mixed-initiative setting.

*Sponsored by the European Research Office of the US Army under grant number W911NF-08-1-0041. The authors' organizations and research sponsors are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

1 Introduction

The classical planning problem [Ghallab *et al.*, 2004, section 2.3] has been extended in a number of ways. One of these extensions is the *distributed (multi-agent) planning problem* [Durfee, 1999]. In fact there is more than one problem that can be described by this term. In this paper, we will add yet another aspect to the problem by requiring that a plan which is a solution to the planning problem must be an *agreed* plan [Wickler, 2008].

The application that motivates this problem is crisis response and management. The scenario here is that a number of agencies want to provide relief after a disaster has struck. These agencies usually have different (but overlapping) capabilities and no authority over each other. Truthfulness and a collaborative attitude are assumed in this scenario.

1.1 The Collaborative Multi-Peer Planning Problem

The distributed planning problem here is distributed in the sense that both, the agents creating the plan and the agents executing the plan are distributed. More specifically, we consider a group of agents that has a set of goals for which they must find a plan and which assigns actions in the plan to agents for execution. The initial state and the

goals are assumed to be known and the same for all agents. It is assumed that all agents want to collaborate to achieve the common goals.

Furthermore, agents may have different capabilities in which case the assignment of actions must be to agents that are capable of performing them. Capabilities may be overlapping, meaning actions can be performed by more than one agent, thereby making the assignment of actions to agents a non-trivial task.

Another important issue is authority. While there may be agents that have authority over other agents and thus can assign actions to those agents, we will not assume that there is a single agent that has authority over all others. This means the planning problem cannot be solved by having one agent plan for all others. Only those agents with authority over other agents will be allowed to plan for them, and in that case the problem can be reduced to one in which only the sub-group of agents that (between them) have authority over all others need to be considered, where this sub-group is a group of peers.

1.2 Agreed Plans

Note that it is not difficult to solve the above problem as a classical planning problem. That is, it is simple enough to find a sequence of actions that transforms the initial state into a goal state. However, for the problem instances we envisage every solution must involve more than one agent and since no agent has authority over any other, no agent can come up with a plan that it can ensure will be executed. Peers may refuse the activities assigned to them, or multiple agents may want to execute the same activities that are required to get to a given goal state.

The heart of the distributed planning problem is therefore not the finding of a plan, but the finding of an *agreed* plan. Informally, by an agreed plan we mean a classical plan in which for every action in the plan there is an agent to which this action is assigned. Furthermore, this agent must *commit* to the actions assigned to it in the plan, that is, the agent intends to execute these actions and takes on an obligation to its peers to execute them. Achieving this commitment is usually not considered part of the classical planning problem, but for the planning problem with multiple peers it is vital.

Figure 1: The Scenario Island (Pacifica)

1.3 The Example Scenario

The approach presented in this paper is being verified in a simulated environment with reasonably realistic features. Figure 1 shows a basic map of this environment, an island inspired by the Pacifica suite of scenarios[1]. On the island, there are cities and a network of roads connecting them, but off-road movement is also enabled. There are also several seaports and airports. The actors are of several unit types (ground, armored, aerial or sea units), and may be civilians or non-friendly units. The latter do not play a role in the kind of scenario we envisage here, but have been used in other research that uses the same environment.

The general scenario is located on this island with a potentially non-friendly environment and limited information visibility and sharing. Due to this, the environment provides non-deterministic behavior from the single agent point of view. In the specific scenario we are interested in here, there are heterogenous independent self-interested agents that adopt the shared/joint goals. To fulfill the desired strategic goals in this environment, the agents provide complex capabilities on several levels of planning and control.

Strategic plan generation is provided by a set of *commander agents* that are responsible for each type of *field unit* (ground, aerial, sea and armored) over which they have authority. The number and

[1] http://www.aiai.ed.ac.uk/oplan/pacifica

specialization of commanders reflects the desired scenario setting. The field units are dedicated to a particular commander and receive their tasks from this commander. The hierarchical structure of tactical planners then creates tactical plans for each field unit (tactical planners are part of each unit's tactical layer). Tactical plans are confronted with the multi-agent simulation developed for the environment and adapted to the actual feedback provided by the simulator in real-time. Execution of the plan by the individual unit is simulated and integrated with the environment feedback from the simulation engine.

There are a number of heterogeneous agent types operating in the scenarios, e.g.:

- **Commanders**: abstract units (not geographically situated) that represent *Ground*, *Armored*, *Aerial* and *Sea* headquarters.
- **Stationary units**: such units include:
  - *Cities* which provide assembly points for civilians;
  - *Supply depots* and *Petrol pumps* which provide resources of various types;
  - *Airfields* which are landing and refueling zones for aerial units; and
  - *Seaports* which are docking and refueling zones for sea units.
- **Mobile units**: agents with the ability to move on the island, including:
  - *Transporters*: ground units, which are can provide transportation of other unit(s), material or civilians;
  - *Construction units*: can repair damages or assemble/disassemble stationary units;
  - *Medical units*: provides medical care for other units or some rescue operations;
  - *Armored units*: for the protection of other units or to secure an area or convoy;
  - *Aerial units*: UAVs with an extended visibility range; and
  - *Sea units*: for transportation over water.

2 <I-N-C-A> and Task-Centric BDI

In this section we will formally define the concepts that form the basis for our definition of the collaborative multi-peer planning problem and, more importantly, what constitutes a solution to this problem. We will begin with activities that form the basic ingredients of a plan. Based on this we will define activity networks as objects in the <I-N-C-A> ontology [Tate, 2003]. Activity refinements provide a way of breaking an activity down into sub-tasks. This will provide a fairly standard HTN planning framework [Sacerdoti, 1975; Tate, 1977]. To extend this to the collaborative multi-peer planning problem and finding agreed plans we need to define mental states of agents and we shall do so using a BDI approach [Rao and Georgeff, 1991]. Specifically, we will define different types of beliefs that agents have about the world and each other. Finally, we will briefly discuss the concept of capabilities as this is important for the planning as well as the mental states of agents.

2.1 Activities

Activities can be defined in a number of ways depending on what needs to be expressed in the planning domain. To ground our definition of the collaborative multi-peer planning problem we shall adopt a simple, STRIPS-like activity model. Other more expressive representations may be used here, e.g. ADL [Pednault, 1989]. What we mean by an activity schema corresponds to a STRIPS operator.

An *activity schema* consists of three components:

- s: the signature of the activity schema $n(v_1, \ldots, v_k)$ where
  - n is the unique name of the activity schema and
  - $v_1, \ldots, v_k$ are variables representing the parameters of the activity schema;
- C: the set of preconditions of the activity schema which can be literals in first-order logic or state variable assignments;
- E: the set of effects of the activity schema which can be literals in first-order logic or state variable assignments.

By an *activity* we mean a partially instantiated activity schema. That is, some of the variables representing parameters may be replaced by symbols referring to objects in the domain and there may be more than one instance of the same activity schema in a plan. We can use the signature plus a unique label to refer to activities.

By an *action* we mean an activity that is fully instantiated, i.e. all the parameters are ground.

If o is an activity schema then we use $o.s$ to refer to the signature of the schema, $o.C$ to refer to the preconditions of the schema, and $o.E$ to refer to the effects of the schema. This notation disambiguates the meaning of overloaded symbols. In general, we will use lower case letters for components that are objects of some kind whereas upper case letters are used for components that are sets of things.

For example, in the scenario described above, a simple activity schema could represent the transporting of some item from one location to another as follows:

```
(:activity-schema
        :signature (transport ?item ?loc1 ?loc2)
        :preconditions (at ?item ?loc1)
        :effects (at ?item ?loc2) ¬(at ?item ?loc1))
```

An action (e.g. in a plan) would then fully specify the parameters, e.g. `(transport food12 city34 depot56)`. Preconditions and effects are inherited from the activity schema with the given action name.

2.2 Activity Networks: <I-N-C-A>

Activities are organized in networks to form plans. We consider such a network to consist of 4 components described by an <I-N-C-A> object:

- I: the set of issues in the activity network, e.g. flaws or opportunities;
- N: the set of activities in the network (at all levels of refinement);
- C: the set of constraints associated with the network, e.g. ordering, time or resources;
- A: a set of annotations, e.g. rationale.

If p is an <I-N-C-A> activity network we use $p.I$, $p.N$, $p.C$, and $p.A$ to refer to the issues, activities, constraints, and annotations of this plan respectively. A more detailed and formal description of the <I-N-C-A> framework can be found in [Wickler *et al.*, 2006].

2.3 Activity Refinements

Refinements are used to break down high-level activities that are part of an <I-N-C-A> activity network into (primitive) actions in HTN planning. An activity refinement consists of the following components:

- n: the unique name of the refinement;
- a: the signature of the activity to be refined (also called the task);
- A: the set of sub-activities that constitute the refinement;
- C: some constraints on activities in A.

If there is no refinement available for a given activity then we assume that this activity is *primitive* and there should be an agent that can execute this activity. If there is a refinement available for a given activity we consider this activity *abstract* and it needs to be refined before it becomes executable.

For example, in the scenario described above, the following simple refinement could be used to specify how to accomplish the task of moving an item from one location to another:

```
(:refinement move-by-truck
        :signature (transport ?item ?loc1 ?loc2)
        :activities (
                1 (load ?item ?truck)
                2 (drive ?truck ?loc1 ?loc2)
                3 (unload ?item ?truck))
        :constraints
                (at ?truck ?loc1)
                (ordering 1 2 3))
```

2.4 Beliefs Desires and Intentions

Activities, activity networks, and activity refinements together form an HTN-style planning framework that does not treat agents in any special way. For the collaborative multi-peer planning problem we need to define the internal structure of an agent in more detail (see also [Wickler *et al.*, 2007]) and

we shall assume a BDI-like framework here. That is, the mental state of an agent can be described by a set of beliefs ($\mathcal{B}$), desires ($\mathcal{D}$), and intentions ($\mathcal{I}$).

Both, desires and intentions, are represented as <I-N-C-A>-objects, which are plans in our approach. This is simply to fit in with our HTN planning framework. Desires include all those activities that an agent would like to perform at some point in the future. These activities can be abstract or primitive, and they may or may not executable.

Intentions include activities that an agent has scheduled for execution. Non-primitive activities must be refined by the planner until all intentions are primitive activities and the agent believes them to be executable (when they are due to be executed). Intentions need not be immediate, that is, they are usually scheduled for an execution time in the future.

An agent may also be committed to performing some of its intentions which are activities in our task-centric view. Thus, in addition to the usual mental attitudes defined in BDI, an agent has a set of commitments $\mathcal{C}$ where each commitment consists of the following components:

- $a \in \mathcal{I}.N$: the activity the agent has pledged to perform;
- p: the precondition under which the agent has pledged to undertake a;
- A: the set of agents to which the pledge was made; and
- Λ: a set of decommitment rules, where each $\lambda \in \Lambda$ consists of:
  - ρ the precondition under which the agent is allowed to drop the intention a;
  - ψ the alternative plan that the agent will adopt instead of a.

The commitment precondition p reflects the preconditions $a.C$ for the applicability of a as an activity (as defined above). Should an agent believe in a decommitment precondition ρ of one of the decommitment rules, it can drop the activity a from its current set of intentions and adopt the postcondition ψ of the selected decommitment rule.

2.5 Belief Types

To define the collaborative multi-peer planning problem, the beliefs of an agent must include various types of beliefs about the activities that can be performed by the different agents defined in the planning problem. Thus, we shall divide the beliefs into the following components:

- S: the usual knowledge about the world state, which will mostly be used to verify the preconditions for activities and refinements are satisfied;
- C: the set of capability descriptions known to the agent, i.e. a set of pairs (a, o) where a is the agent that has advertised the capability and o is the signature of an activity schema that describes the advertised capability;
- R: the set of activity refinements known to the agent;
- P: the set of primitive activities that this agent can execute (and knows that it can).

For example, the commander in charge of sea units could advertise the capability to transport items from one location to another. Additional constraints may be given with a capability advertisements, such as the fact that the two locations in question must both be sea ports.

The refinements R would be a set of refinements as described above. The set of primitive activities are different for the agents in our example as each agent only needs to know the activities it itself can execute.

For example, a truck agent knows that it can drive from one location to another if there exists a road that connects the two locations. It may still require some form of planning to work out more detail, but the agent must know that it can always execute this action if the preconditions are satisfied.

2.6 Capabilities

The notion of capabilities described so far is actually too simplistic for practical applications. It is assumed that capabilities are described using a shared ontology of activity descriptions. Since an agent associates only the signature of an activity with another agent in a capability description,

there needs to be a description associated with that signature that all agents agree on. A hierarchical ontology of activities would be more useful for this purpose than the simple model described above.

Also, it is not realistic to assume that an agent will advertise as part of its capability description all the constraints that it considers preconditions for the application of this capability. For example, one precondition that applies to every capability description is that the agent is not already otherwise committed. That is, agents that can only perform one action at a time may in principle be capable of performing an advertised capability, but they may not be able to perform it at any requested time. As intentions and commitments are expected to change relatively frequently but advertised capabilities are expected to remain constant, such scheduling constraints should not be part of a capability advertisement.

Another practical concern is that agents usually only perform their advertised capabilities as part of a larger protocol which in itself can be seen as a plan. For example, an agent might expect to be paid for performing its capability and there may be actions that correspond to that process that the capability requester needs to perform.

3 Formalizing the Collaborative Multi-Peer Planning Problem

We can now formalize the collaborative multi-peer planning problem. In addition to the traditional components of a planning problem (operators, initial state, and goal) this requires a set of agents with their mental (BDI) states as described above. The next step must then be a definition of what constitutes a solution to such a problem. This will define the semantics of the problem.

3.1 The Problem Specification

A collaborative multi-peer planning problem is given by a pair (A, N) where A describes a set of agents and N is an initial activity network.

The first component, $A = \{a_1, \ldots, a_n\}$, consists of a set of agent descriptions where each agent a_i is defined by its beliefs $a_i.\mathcal{B}$, its desires $a_i.\mathcal{D}$, its intentions $a_i.\mathcal{I}$ and its commitments $a_i.\mathcal{C}$ as described above. Note that the beliefs include beliefs about the world state, the capabilities of the agent itself as well as other agents, available refinements for breaking down tasks, and primitive activities the agent can execute. Having refinements as part of the agents' knowledge means they do not need to be listed explicitly as part of the planning problem.

The second component N is an <I-N-C-A> activity network consisting of issues $N.I$, activities (nodes) $N.N$, constraints $N.C$, and annotations $N.A$. In this HTN-style specification of a planning problem it is not necessary to list the initial state and the goal separately as they can be defined in the initial network with two dummy activities `init` and `goal` as is usual in HTN planning.

We shall assume that there are no inconsistencies in the planning problem and that all agents have complete knowledge of each other's capabilities. The former implies that the initial state implicit in N and the agents' beliefs about the current world state $a_i.\mathcal{B}.S$ are equivalent. The latter implies that for all agents a_i, a_j their beliefs about capabilities are the same, i.e. $a_i.\mathcal{B}.C = a_j.\mathcal{B}.C$.

3.2 Solutions

The above definition of a collaborative multi-peer planning problem is not significantly different from classical HTN planning, except that the agents that execute actions are made explicit. Note that this already creates a different solution space if one assumes that agents can only execute one action at a time.

The solutions we are looking for must include an important additional condition: they must be agreed by all the agents in the problem. This can be formalized by requiring that every action that is part of the plan and is assigned to an agent must be an element of that agent's intentions, i.e. every agent is committed to executing its share of the plan as a result of the planning process.

An agreed solution to a collaborative multi-peer planning problem (A, N) is a pair (N', s) where:

- N' is a ground, primitive activity network and
- s is an assignment of activities in N' to agents in A

such that:

- there exists a decomposition tree Δ from N to N'
- every action is assigned to an agent, i.e. $\forall a \in leafs(\Delta) : s(a) \in A$
- assignments are to agents capable of performing the activities, i.e. $\forall a \in leafs(\Delta) : a \in s(a).\mathcal{B}.C$
- agents are committed to performing activities assigned to them, i.e. $\forall a \in leafs(\Delta) : a \in s(a).\mathcal{I} \wedge$ $\exists c \in s(a).\mathcal{C} : c.a = a \wedge c.p = a.C \wedge c.A = A$

In order to make possible the execution of a collection of distributed plans that constitute the solution to a multi-peer planning problem we need to make the participating agents not only agree with execution of the respective plans but to commit to their execution. Unlike agreement, a mere expression of conformity with the proposed plans, the commitment is a richer knowledge structure representing agents attitude to the agreed piece of execution. While the intention to perform an action is somewhat equivalent to agents agreement to perform an action, the commitment also specifies under which circumstances the agent can get decommitted from its commitment, thereby opening up a genuinely larger solution space for the collaborative multi-peer planning problem.

4 An Approach

We will now discuss a solution approach to the collaborative multi-peer planning problem which draws some parallels to the idea behind auctions as a method for solving the problem of finding an *agreed* price.

4.1 Auctions and Agreement

One way of viewing an auction is as a process for *agreeing* on something. A group of agents want to buy an item one of them has to sell, and they need to agree who will buy the item and at what price. To do so, they hold an auction. That is, they first agree on a process they will follow and that will result in the agreed price and buyer. Then they execute that process and have the desired agreement at which point the buyer and seller are committed. Thus, instead of directly seeking the agreement they are after, they first agree on a process that leads to the agreement, and then following this process leads to commitments.

4.2 Finding Agreed Plans

The question then is what such a process could look like in the case of the collaborative multi-peer planning problem. One possible procedure is based on the idea of each agent solving the overall problem as if it had authority over the other agents, then negotiating additional constraints that may not have been part of the capability descriptions, and finally voting for the agreed plan if necessary:

repeat
    every agent:
        solve HTN planning problem
        suggest plan that self would agree to
    for every plan:
        every agent:
            add constraints on own activities
            add decommitment rules on own activities
    every agent:
        modify suggested plan with new constraints
until no agents add further constraints
vote for solution plan
every agent:
    commit to the agreed plan

The agents start by solving the planning problem as a classical planning problem, treating all capabilities as available primitive activities. This may seem like a waste of computational resources but it is unlikely that agents will accept other agents' plans without thinking about the way in which the common goal can be achieved themselves. Furthermore, HTN problems tend to be solvable without too much effort for most practical domains for which there is a known set of standard operating procedures. It is assumed here that agents all want to accomplish the common goal, but every agent may have different criteria for what they consider the *best* solution, which is why they may come up with different plans at this stage.

In the next step agents can suggest the solutions they would like to see implemented to the other

agents. This can be done by using a shared data-structure such as a blackboard, or by means of agents requesting each other to provide solutions peer-to-peer. There is no requirement for every agent to suggest a plan; they may just rely on others, e.g. if there is a peer they fully trust. In the worst case there could be no suggested plans and this simply reflects the fact that there might not be any solution plans to the classical problem, just like there might not be an agreeable price for an auction. Assuming that there are n peers involved in the planning, there may now exist (up to) n initial candidate plans representing alternative solutions at this point in the algorithm.

4.3 Adding Constraints

Once the plans have been suggested, agents can constructively critique each others candidate plans. This can be done by adding further constraints to the activities in the plan. For example, an agent may be assigned an activity that conforms with its advertised capabilities as part of some other agent's plan, but it may not be able to perform the capability at that time or in the specific circumstances. Adding such a constraint at this stage will invalidate the plan and the agent who suggested the plan may then modify its candidate solution to take into account any additional constraints that have been added. The agent may wait with the modification until all peers that are assigned actions in the plan have had an opportunity to add constraints. The modification of the candidate solution may be a minor fix or a completely different plan, but it has to be a valid solution to the planning problem including all the new constraints.

4.4 Adding Decommitment

Instead of constraints that invalidate a plan, an agent may add decommitment rules to activities assigned to it in another agent's candidate solution. These rules do not invalidate the plan but specify how it might be changed at execution time, given certain conditions. Just like the activities in the plan, these decommitment rules need to be agreed by all peers.

There is a number of different decommitment rule types suggested for adoption:

- **Full decommitment:** The basic decommitment strategy is dropping the commitment without any further task to be accomplished. Under defined circumstances the agent is completely released from the commitment.
- **Delegation:** By using this type of the decommitment rule the agent shall be able to find some other agent who will be able to complete its commitment on the original agent's behalf. It is possible that such a commitment will contain unbound variables representing the need to search for an agent suitable for delegation. The basic idea is to find an agent that is able to undertake the commitment under circumstances when the decommitment condition (which is true in case of the original agent) became false, so the new agent is able to fulfill the commitment. The delegated commitment can contain a new set of decommitment rules.
- **Relaxation:** Relaxation is a special decommitment, where the original commitment is replaced with a new commitment with relaxed properties of the task. Such decommitment rule is often used when the agent needs to speculate about its potential inability to provide request quality of service, delivery time or costs. It has been shown that planning with relaxation decommitments rules can contribute to an increased efficiency of coordinated actions in dynamic environments [**?**].

Since decommitment rules do not invalidate the suggested plan, there is no need for the suggesting agent to modify its candidate in response to a decommitment rule being added. However, the suggester may still decide to do so if it believes another plan to be now preferable.

4.5 Voting

This process is continued until no agent adds further constraints or decommitment rules to any of the plans, in which case the set of candidate plans is hopefully not empty. Note that each of these plans represents a valid classical solution. Now the agents need to choose one of these solutions and this can be achieved by voting, provided that each agent is able to construct a preference ordering of the plans. This preference ordering would be linked

with utility functions representing cost and/or benefits that each plan represents for the respective agent. Note that at this stage all the plans available for voting are plans the participating agents are happy to adopt and thus the voting is simply a means for selecting one of the plans. Various voting protocols (e.g. plurality protocol, binary protocol, Borda protocol) can be adopted here.

If the voting process terminates successfully, the selected plan is an abstract solution to the multi-agent planning problem. As the voting process selects the plan from the constrained candidate plans, each annotated with particular decommitment rules, all involved agents commit to the selected plan.

5 Conclusions

This paper defines a new kind of distributed, multi-agent planning problem, the collaborative multi-peer planning problem. In this problem a group of peers (agents without authority over each other) have to agree on a plan that achieves a common goal. Agents are defined with BDI-style mental states, and a plan is considered agreed if the actions in the plan are assigned to agents that are committed to executing these actions. This commitment is an important aspect that needs to be part of the outcome of the planning process. The proposed approach introduces a level of indirection into the planning process—instead of agreeing the plan directly the peers agree on the procedure for finding the plan and then execute this procedure. The resulting plan may include decommitment rules for certain activities in the plan, giving agents that participate in the execution a certain degree of flexibility.

This approach is particularly relevant for coalition operations in which participants in the coalition are peers and the lack of authority over each other adds a new source of problems to the planning process.

The proposed planning algorithm is very efficient, but it is not aimed at fining an optimal solution plan. Agents get committed to pre-selected plan that is presumably optimal (to some degree) according to the suggesting agent's criteria, but more importantly, it should at least acceptable to all agents involved. There may be potential for an improvement of e.g. stability of the plan or longer term use of resources should the agents have the opportunity to choose between quality of the provided solution and various offered decommitment strategies.

References

[Allen *et al.*, 1990] James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning*. Morgan Kaufman, 1990.

[Durfee, 1999] Edmund H. Durfee. Distributed problem solving and planning. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 3, pages 121–164. The MIT Press, 1999.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning*. Morgan Kaufmann, 2004.

[Pednault, 1989] Edwin Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. 1st International Conference on Knowledge Representation and Reasoning (KR)*, pages 324–332. Morgan Kaufmann, 1989.

[Rao and Georgeff, 1991] Anand Rao and Michael Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. 2nd International Conference on Knowledge Representation and Reasoning (KR)*, pages 473–484. Morgan Kaufmann, 1991.

[Sacerdoti, 1975] Earl D. Sacerdoti. The nonlinear nature of plans. In *Proc. 4th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 206–214. Morgan Kaufmann, 1975. Reprinted in [Allen *et al.*, 1990, pages 162–170].

[Tate, 1977] Austin Tate. Generating project networks. In *Proc. 5th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 888–893. Morgan Kaufmann, 1977. Reprinted in [Allen *et al.*, 1990, pages 291–296].

[Tate, 2003] Austin Tate. <I-N-C-A>: A shared model for mixed-initiative synthesis tasks. In Gheorghe Tecuci, editor, *Proc. IJCAI Workshop*

on Mixed-Initiative Intelligent Systems, pages 125–130, 2003.

[Wickler *et al.*, 2006] Gerhard Wickler, Stephen Potter, and Austin Tate. Recording rationale in <I-N-C-A> for plan analysis. In Lee McCluskey, Karen Myers, and Biplav Srivastava, editors, *Proc. ICAPS Workshop on Plan Analysis and Management*, pages 5–11, 2006.

[Wickler *et al.*, 2007] Gerhard Wickler, Stephen Potter, Austin Tate, Michal Pěchouček, and Eduard Semsch. Planning and choosing: Augmenting HTN-based agents with mental attitudes. In *Proc. International Conference on Intelligent Agent Technology*, 2007.

[Wickler, 2008] Gerhard Wickler. Finding agreed plans. In Ruth Aylett and Yvan Petillot, editors, *Proc. 27th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG)*, pages 137–142, 2008.