Planning and Defeasible Reasoning *

Diego R. Garcia
Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur – (8000) Bahia Blanca.
drg@cs.uns.edu.ar

Alejandro J. Garcia
Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur – (8000) Bahia Blanca.
ajg@cs.uns.edu.ar

Guillermo R. Simari
Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur – (8000) Bahia Blanca.
grs@cs.uns.edu.ar

ABSTRACT

We present an argumentation-based formalism that an agent could use for constructing plans. We will analyze the interaction of arguments and actions when they are combined to construct plans using Partial Order Planning techniques.

Categories and Subject Descriptors

I.2.4 [**Artificial intelligence**]: Knowledge Representation Formalisms and Methods; I.2.8 [**Artificial intelligence**]: Plan execution, formation, and generation

General Terms

Algorithms, Theory

Keywords

Agents, Planning, Argumentation

1. INTRODUCTION

We present an argumentation-based formalism an agent could use for constructing plans using partial order planning techniques. In our proposed approach, actions and arguments will be combined to construct plans. As we will explain next, actions preconditions can be satisfied by other actions effects (as usual) or by conclusions supported by arguments based on inference rules and other actions effects.

When actions and arguments are combined in a partial order plan, new types of interferences appear (called threats in [2]). These interferences need to be identified and resolved in order to obtain valid plans. Since our interest here lies in exploring the important issues that need to be addressed, the main contribution will be to show meaningful examples and the description of the new type of interferences.

*Partially supported by SGCyT U. N. del Sur, CONICET and Agencia Nacional de Promoción Cientfica y Tecnológica.

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

2. ARGUMENTATION AND ACTIONS

In this section, we introduce a brief description of a formalism that combines actions and defeasible argumentation based on a previous work reported in [5, 4]. Our formalism follows the logic programming paradigm for knowledge representation called Defeasible Logic Programming (DeLP) [1]. Thus, the agent's knowledge will be represented by a DeLP program and the agent will be able to perform defeasible reasoning over this knowledge.

The agent's knowledge base will be a defeasible logic program $\mathcal{K} = (\Psi, \Delta)$, where Ψ should be a consistent set of *facts*, and Δ a set of *defeasible rules*. Defeasible Rules are denoted $L_0 \prec L_1, \ldots, L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals. A defeasible rule "$Head \prec Body$" is the key element for introducing *defeasibility* [3]. *Strong negation* "$\sim$" can appear in the head of defeasible rules, and it could be used to represent conflicting information. In DeLP arguments for conflicting pieces of information are build and then compared to decide which one prevails. A definition of argument adapted from [1] follows:

Definition 1 [Argument]
Let L be a literal, and $\mathcal{K} = (\Psi, \Delta)$ a defeasible logic program. We say that $\langle \mathcal{A}, L \rangle$ is an argument for L if $\mathcal{A}$ is a minimal, non-contradictory, set of defeasible rules of Δ, such that $\Psi \cup \mathcal{A} \vdash L$.

DeLP provides a mechanism for deciding which argument prevails and therefore, which literals are warranted (see [1] for further details). Besides its knowledge base $\mathcal{K}$, an agent will have a set of actions Γ that it may use to change its world. We recall the formal definitions introduced in [4, 5].

Definition 2 [Action] An action A is an ordered triple $\langle \mathsf{X}, \mathsf{P}, \mathsf{C} \rangle$, where X is a consistent set of literals representing consequences of executing A, P is a set of literals representing preconditions for A, and C is a set of constraints of the form *not* L, where L is a literal. We will denote actions as follows: $\{X_1, \ldots, X_n\} \xleftarrow{A} \{P_1, \ldots, P_m\}, not\ \{C_1, \ldots, C_k\}$, where *not* $\{C_1, \ldots, C_k\}$ represents $\{not\ C_1, \ldots, not\ C_k\}$.

Accordingly, the condition that must be satisfied before an action $A = \langle \mathsf{X}, \mathsf{P}, \mathsf{C} \rangle$ can be executed contains two parts: P, which mentions the literals that *must* be warranted, and C, which mentions the literals that *must not* be warranted.

Definition 3 [Applicable Action] Let $\mathcal{K} = (\Psi, \Delta)$ be an agent's knowledge base. Let Γ be the set of actions available to this agent. An action A in Γ, is applicable if every

978-81-904262-7-5 (RPS)

precondition P_i in P has a warrant built from (Ψ, Δ) and every constraint C_i in C fails to be warranted.

It is clear that only applicable actions can be executed. When an action is applied by the agent, it's effect change the environment and the set $\mathcal{K}$. The effect of the execution of an applicable action in our formalism is defined below:

Definition 4 [Action Effect] Let $\mathcal{K} = (\Psi, \Delta)$ be an agent's knowledge base. Let Γ be the set of actions available to this agent. Let A be an applicable action in Γ defined by:

$$\{X_1, \ldots, X_n\} \xleftarrow{A} \{P_1, \ldots, P_m\}, not\ \{C_1, \ldots, C_k\}$$

The effect of executing A is the revision of Ψ by X, i.e. $\Psi^{*\mathsf{X}} = \Psi^{*\{X_1,\ldots,X_n\}}$. Revision will consist of removing any literal in Ψ that is complementary of any literal in X and then adding X to the resulting set. Formally:

$$\Psi^{*\mathsf{X}} = \Psi^{*\{X_1,\ldots,X_n\}} = (\Psi - \overline{\mathsf{X}}) \cup \mathsf{X}$$

where $\overline{\mathsf{X}}$ represents the set of complements of members of X.

The argumentation formalism described above allows an agent to represent knowledge about the environment and to define the actions it can perform. However, it does not describe how to construct a plan to achieve the agent's goals.

A simple formulation of a planning problem defines three inputs: a description of the *initial state* of the world in some formal language, a description of the agent's *goal*, and a description of the possible *actions* that can be performed. The initial state is the agent's current representation of the world, and in our case it will be the set Ψ. The agent's goals will be represented as a set G of literals. The agent will satisfy its goals when through a sequence of actions it reaches some state Ψ' where each literal of G is warranted.

In the following section, we will describe how partial order planning techniques can be combined with the described formalism to provide the agent with the ability to build plans.

3. ARGUMENTATION IN PARTIAL ORDER PLANNING

The basic idea behind a regression Partial Order Planning (POP) algorithm [2] is to search through plan space. The planner starts with an initial plan that consists solely of a *start* step (whose effects encode the initial state conditions) and a $finish$ step (whose preconditions encode the goals) (see Figure 1(a)). Then it tries to complete the initial plan by adding new steps (actions) and constraints until all step's preconditions are guaranteed to be satisfied. The main loop in a traditional POP algorithm makes two type of choices:

- *Supporting unsatisfied preconditions*: Chooses a step to achieve a selected unsatisfied precondition.
- *Resolve threats*: If a step might possibly interfere with the precondition being supported by another step, it chooses a method to resolve this *threat*.

To combine the argumentation formalism with POP, we must consider the use of arguments for supporting unsatisfied preconditions, besides actions.

The following definitions are introduced for identifying different sets of literals present in an argument, that will be considered when a plan is constructed.

Definition 5 [Heads-Bodies-Literals] Given an argument $\langle B, h\rangle$, $heads(\mathcal{B})$ is the set of all literals that appear as heads of rules in $\mathcal{B}$. Similarly, $bodies(\mathcal{B})$ is the set of all literals that appear in the bodies of rules in $\mathcal{B}$. The set of all literals appearing in $\mathcal{B}$, denoted $literals(\mathcal{B})$ is the set $heads(\mathcal{B}) \cup bodies(\mathcal{B})$.

Definition 6 [Argument base] Given an argument $\langle B, h\rangle$ we will say that the base of $\mathcal{B}$ is the set $base(\mathcal{B}) = bodies(\mathcal{B}) - heads(\mathcal{B})$.

Definition 7 [Conclusion] Given an argument $\langle B, h\rangle$ we will say that the conclusion of $\mathcal{B}$ is the literal $conclusion(\mathcal{B}) = heads(\mathcal{B}) - bodies(\mathcal{B})$.

Following, we will present an example to illustrate how traditional POP algorithm can be extended to consider arguments as planning steps. For simplicity, we present a propositional problem that defines actions without constraints.

Example 1 Consider an agent that works at night and its job is cleaning rooms in a building. The agent arrives to a room where the light switch is set to off and has to build a plan for having that room cleaned. The agent have the following knowledge base: $\Psi = \{switch\_off\}$ and

$$\Delta = \left\{ \begin{array}{c} light\_in\_room \prec switch\_on \\ \sim light\_in\_room \prec switch\_on, \sim electricity \end{array} \right\}$$

The agent's goal is $G = \{room\_clean\}$, and the available actions are:

$$\{room\_clean\} \xleftarrow{clean\_room} \{light\_in\_room\}, not\ \{\}$$
$$\{switch\_on, \sim switch\_off\} \xleftarrow{turn\_switch\_on} \{switch\_off\}, not\ \{\}.$$

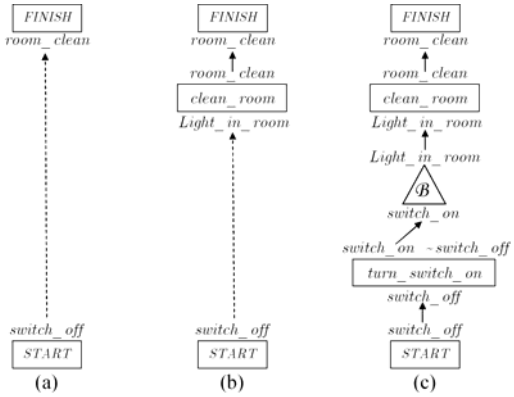

Figure 1: Different partial plans for Example 1.

Figure 1(a) shows the initial plan for example 1. Figure 1(b) depicts an incomplete plan where only actions were considered. Figure 1(c) shows a complete plan obtained using actions and arguments.

In our approach, we will distinguish between two types of steps: *action steps* (i. e. steps that represent the execution of an action) and *argument steps* (i. e. arguments used in the plan to support the precondition of some action step). *Action steps* are depicted by square nodes labelled with the action name. The literals that appear below an action step represent the preconditions of the action step, and the literals that appear above represent its effects. The solid arrows in the figure represent *causal links* and dashed arrows represent *ordering constraints*. Finally, triangles represent *argument steps* and are labelled with the argument name. The literal at the top of the triangle is the conclusion of the argument (Definition 7), and the literals at the base of the triangle represent the base of the argument (Definition 6).

In Figure 1(a) there is only one unsatisfied precondition *room_clean*, and the only possible way to satisfy it is by the action *clean_room*. A new step *clean_room* is added

(Figure 1(b)) to the plan and it's precondition *light_in_room* becomes a new unsatisfied subgoal. Observe that none of the actions available achieve *light_in_room*, then it is not possible to obtain a plan if only actions are considered.

However, from the rules Δ of the agent's knowledge base it is possible to construct the (potential) argument $\mathcal{B}$=$\{$ $(light\_in\_room \prec switch\_on)$ $\}$ that supports *light_in_room*. Therefore, an alternative way to achieve *light_in_room* would be to use $\mathcal{B}$ for supporting *light_in_room*, and then to find a plan for satisfying all the literals in the base of $\mathcal{B}$ ($base(\mathcal{B}) = \{switch\_on\}$). Figure 1(c) shows this situation. The argument $\mathcal{B}$ is chosen to support *light_in_room* and the literal *switch_on* becomes a new subgoal of the plan. Then, the action *turn_switch_on* is selected to satisfy *switch_on* and the corresponding step is added to the plan. The precondition $switch\_off$ of the step *turn_switch_on* is achieved by the *start* step, so a causal link is added and a plan is obtained.

Note that $\mathcal{B}$=$\{$ $(light\_in\_room \prec switch\_on)$ $\}$ is a "*potential argument*" because it is conditioned to the existence of a plan that satisfies its base. This argument can not be constructed from a set of facts, as usual. The reason is that at the moment of the argument construction it is impossible to know which literals are true, because they depend on steps that will be chosen later in the planning process. A formal definition of potential argument follows:

Definition 8 [Potential Argument]
Let L be a literal, and Δ a set of defeasible rules. We say that $\langle\langle \mathcal{A}, L\rangle\rangle$ is a potential argument for L if $\mathcal{A}$ is a minimal, non-contradictory, set of defeasible rules of Δ, such that $base(\mathcal{A}) \cup \mathcal{A} \vdash L$

Another thing to consider is that the existence of this argument $\mathcal{B}$ is not enough to have a warrant for *light_in_room*, because $\mathcal{B}$ could be defeated by a counter-argument (for example, when there is no electricity in the building). Recall that in order to be able to apply an action, all it's preconditions has to be warranted.

4. INTERFERENCES AMONG ACTIONS AND ARGUMENTS

When only actions are considered, there is only one type of destructive interference that can arise in a plan. In the traditional POP algorithm, this interference is captured by the notion of *threat* (see Figure 2(a)). When actions and arguments are combined to construct plans, new types of interferences appear that need to be identified and resolved in order to obtain a valid plan. We will extend the notion of *threat* to identify all the different types of interferences.

Figure 2(a) shows the kind of threat that appears in the traditional POP algorithm: the precondition a of A_1, supported by A_2, is threatened by the action step A_3 because it negates a. Note that $\overline{a}$ is an effect of A_3, where $\overline{a}$ stands for the complement of a with respect to strong negation, i. e. $\overline{p}$ is $\sim p$, and $\overline{\sim p}$ is p.

This type of threat involves only action steps and will be called *action-action threat*. However, as we consider also arguments to construct a plan, a different kind of threat could arise involving action steps and argument steps. Consider the situation shown in Figure 2(b). In this case, the action step A_3 threatens the argument step $\mathcal{B}$ because it negates a literal present in the argument $\mathcal{B}$. Note that $\overline{n}$ is an effect of A_3 and $n \in literals(\mathcal{B})$ (see definition 5). The argument step $\mathcal{B}$ was added to the plan to support the precondition

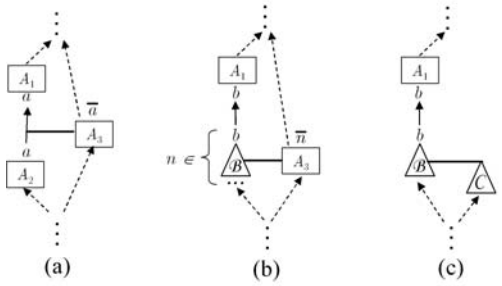

Figure 2: Interferences that could arise in a plan

b of the action step A_1. If A_3 makes $\overline{n}$ true before A_1 is executed, the argument $\mathcal{B}$ will not exist at the moment a warrant for b is needed to execute A_1. This type of threat will be called *action-argument threat*.

Finally, there is another type of threat to consider that involves only arguments. Arguments are introduced in the plan to have a warrant for the precondition of some action step. Since the arguments could be defeated by a counter-argument, the precondition would not be warranted. This situation is shown in figure 2(c): The argument step $\mathcal{B}$ was added to the plan to support the precondition b of the action step A_1. However, at the moment a warrant for b is needed to execute A_1, there exist a defeater $\mathcal{C}$ for $\mathcal{B}$. We will refer to this type of threat as *argument-argument threat*. Observe that the defeater $\mathcal{C}$ is not necessarily an argument step of the plan. The argument $\mathcal{C}$ could be any defeater that can be built from the effects of the actions steps that could be ordered to come before A_1 in the plan.

5. CONCLUSIONS

We have show how an argumentation-based formalism can be combined with partial order planning technics to construct plans. The traditional POP algorithm was extended to consider arguments as planning steps. When actions and arguments are combined to construct plans, new types of interferences appear. We have extended the notion of threat to consider the new types of interferences.

6. REFERENCES

[1] A. J. García and G. R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[2] J. Penberthy and D. S. Weld. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In *In Proc. of the 3rd. Int. Conf. on Principles of Knowledge Representation and Resoning, 113-124.*, 1992.

[3] J. Pollock. *Cognitive Carpentry: A Blueprint for How to Build a Person*. MIT Press, 1995.

[4] G. R. Simari and A. J. García. Actions and arguments: Preliminaries and examples. In *Proceedings of the VII Congreso Argentino en Ciencias de la Computación*, pages 273–283. Universidad Nacional de la Patagonia San Juan Bosco, El Calafate, Argentina, Oct. 2001. ISBN 987-96-288-6-1.

[5] G. R. Simari, A. J. García, and M. Capobianco. Actions, Planning and Defeasible Reasoning. In *In Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR2004)*, pages 377–384, 2004. ISBN. 92-990021-0-X.