Analyzing the Interaction between Actions and Arguments in Partial Order Planning

Diego R. García   Alejandro J. García   Guillermo R. Simari

Laboratorio de Investigación y Desarrollo de Inteligencia Artificial
Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Email: {drg,ajg,grs}@cs.uns.edu.ar

ABSTRACT

This research line involves the study of the interaction of arguments and actions when they are combined to construct plans using Partial Order Planning techniques. When actions and arguments are combined new types of interferences appear in the plans. This interferences need to be identified and resolved in order to obtains valid plans.

Keywords: Defeasible Reasoning, Partial Order Planning.

1 INTRODUCTION

In previous work [2, 6, 5, 4], we have introduced a formalism where agents represent their knowledge about their environment using the language of Defeasible Logic Programming (DeLP) [1], and have a set of actions that they can execute in order to change the environment where they are performing their tasks. In the Appendix A we include a brief description of this formalism.

The agent's knowledge base will be represented by a *defeasible logic program* $\mathcal{P} = (\Psi, \Delta)$, where Ψ should be a consistent set of *facts*, and Δ a set of *defeasible rules*.

A simple formulation of a planning problem defines three inputs [7]: 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. In order to achieve its goals, the agent will start in the initial state Ψ and it will execute a sequence of actions transforming Ψ into 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.

Partially supported by Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) PIP 5050, and Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002, Nro. 13096)

2 ARGUMENTATION IN PARTIAL ORDER PLANNING

The argumentation formalism allow an agent to represent it's knowledge about it's environment. In particular, it define the actions an agent can perform, when an action is applicable and how to compute its effects. However, it does not describe how to construct a plan for achieving the agent's goals. In a previous work [2] we introduce the study of the combination of this formalism with Partial Order Planning techniques in order to provide the agent with the ability to built plans.

The basic idea behind a regression Partial Order Planning (POP) algorithm [3] is to search backwards through plan space instead of state space, as state-based planners do. The planner starts with an initial plan that consist 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 attempts to complete this initial plan by adding new steps (actions) and constrains until all step's preconditions are guaranteed to be satisfied. The main loop in the POP algorithm makes to type of choices:

- Supporting unsatisfied preconditions: all steps effects that could possibly be constrained to unify with the desired proposition are considered. It choose one step nondeterministically and then adds a causal link to the plan to record that the precondition is achieved by the chosen step.
- Resolve threats: If a step might possibly interfere with the precondition being supported by a casual link, it nondeterministically chooses a method to resolve the threat: either by reordering steps in the plan (adding ordering constraints), posting additional subgoals, or by adding new equality constraints.

In our argumentation formalism (see Appendix A), and action is applicable if every precondition of the action has a warrant built from the agent's current knowledge base, and every constraint fails to be warranted. To combine this formalism with POP, we must consider the use of arguments for supporting unsatisfied preconditions, besides actions. The combined use of argumentation and actions to build plans introduces new issues not present in the traditional POP algorithm that need to be addressed. In this work we will focus on analyzing the interaction between actions and arguments.

As we will describe below, arguments can not be constructed from a set of facts, as usual, because at the moment of the argument construction it is impossible to know which literals are true. The following definitions are introduced for identifying this set of literals.

Definition 1 **[Heads-Bodies-Literals]** Given an argument structure $\langle B, h\rangle$, $\mathit{heads}(\mathcal{B})$ is the set of all literals that appear as heads of rules in $\mathcal{B}$. Similarly, $\mathit{bodies}(\mathcal{B})$ is the set of all literals that appear in the bodies of rules in $\mathcal{B}$. The set of all literals a appearing in $\mathcal{B}$, denoted $\mathit{literals}(\mathcal{B})$ is the set $\mathit{heads}(\mathcal{B}) \cup \mathit{bodies}(\mathcal{B})$

Definition 2 **[Argument base]** Given an argument structure $\langle B, h\rangle$ we will say that the base of $\mathcal{B}$ is set $\mathit{base}(\mathcal{B}) = \mathit{bodies}(\mathcal{B}) - \mathit{heads}(\mathcal{B})$.

Definition 3 **[Conclusion]** Given an argument structure $\langle B, h\rangle$ we will say that the conclusion of $\mathcal{B}$ is the set $\mathit{conclusion}(\mathcal{B}) = \mathit{heads}(\mathcal{B}) - \mathit{bodies}(\mathcal{B})$.

Example 1 Given the argument structure $\langle B, b\rangle$ where $\mathcal{B}$= { $(b \prec c, d)$, $(c \prec e)$ }, the corresponding sets are:

$$\begin{array}{lcl}
\mathit{heads}(\mathcal{B}) & = & \{b, c\} \\
\mathit{bodies}(\mathcal{B}) & = & \{c, d, e\} \\
\mathit{literals}(\mathcal{B}) & = & \{b, c, d, e\} \\
\mathit{base}(\mathcal{B}) & = & \{d, e\} \\
\mathit{conclusion}(\mathcal{B}) & = & \{b\}
\end{array}$$

Following, we will present an example to illustrate the POP algorithm and introduce the basic terminology and graphical representation that will be used in the rest of the paper. For simplicity, we present a propositional planning problem that defines actions without constraints.

Example 2 Suppose that an agent has the following knowledge base: $\Psi = \{e, f, h\}$ and Δ= { $(b \prec c, d)$ }. The agent's goal is $G = \{a, g\}$, and the available actions are:

$$\{a\} \xleftarrow{A_1} \{b\}, \mathit{not}\ \{\} \quad \{c\} \xleftarrow{A_2} \{e\}, \mathit{not}\ \{\}$$
$$\{d\} \xleftarrow{A_3} \{f\}, \mathit{not}\ \{\} \quad \{g\} \xleftarrow{A_4} \{h\}, \mathit{not}\ \{\}$$

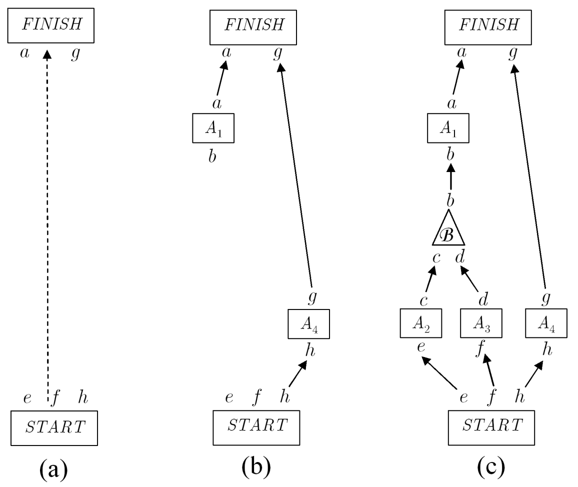

Figure 1: Different partial plans for Example 2

Figure 1 shows different (and possibly incomplete) plans obtained from choosing different alternatives to achieve the unsatisfied preconditions. The square nodes represent action steps. The squares labeled START and FINISH represent the *start* and *finish* steps respectively. The literals that appear below a step represent the preconditions of that step, and the literals that appear above represent its effects. The solid arrows in the figure represent *causal links* and dashed arrows represent *ordering constrains*. Causal links are used to explicity record the source for each proposition during planning. Ordering constraints are used to explicity establish an order between two steps. By definition the *start* step come before the *finish* step and the rest of the steps are constrained to come after the *start* step and before the *finish* step. All causes are constrained to come before their effects, so a causal link also represent an ordering constraint. Finally, the triangles represents arguments. The literal at the top of the triangle is the conclusion of the argument (Definition 3), and the literals at the base of the triangle represent the base of the argument (Definition 2).

Figure 1(a) shows the initial plan. Figure 1(b) shows an incomplete plan where only actions (not arguments) were considered to achieve the unsatisfied preconditions. Initially the are two unsatisfied preconditions: a and g, and the only possible way to satisfy them is by actions A_1 and A_4 respectively. Introducing this two steps add new unsatisfied subgoals b and h (the preconditions of A_1 and A_4). The *start* step achieve h so no new step is needed: a casual link is added. Observe finally that b remains unsatisfied, because none of the actions available achieve this precondition.

However, note that from the rules Δ of the agent's knowledge base it is possible to construct the (potential) argument $\mathcal{B}$={ $(b \prec c, d)$ } that supports b. Therefore, an alternative way to achieve b would be to use $\mathcal{B}$ for supporting b, and then to find a plan for satisfying all the literals in the base of $\mathcal{B}$ ($base(\mathcal{B}) = \{c, d\}$). Figure 1(c) shows this situation. The argument $\mathcal{B}$ is chosen to support b and actions A_2 and A_3 are selected to satisfy c and d respectively. The preconditions of both actions are achieved by the *start* step, so the corresponding causal links are added and a plan is obtained.

Note that $\mathcal{B}$={ $b \prec c, d$ } is a "*potential argument*" because it is conditioned to the existence of a plan that satisfy 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.

3 INTERACTION BETWEEN ACTIONS AND ARGUMENTS

As mentioned before, the classical POP algorithm has to find and resolve *threats* present in the plan, that is, actions that might interfere with the precondition being supported by another action. Figure 2(a) shows a threat: the precondition a of A_2, supported by A_1, is threatened by the action 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.

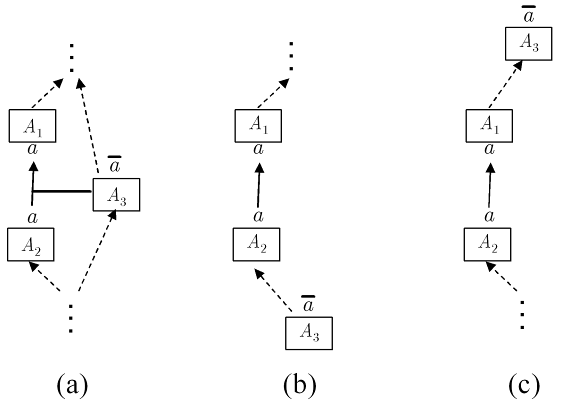

Figure 2: In (a) an action threaten the precondition supported by another action. In (b) the threat is solved by *demotion*. In (c) the threat is solved by *promotion*

The way to resolve this threat is to add an ordering constraint to make sure that A_3 is not executed between A_2 and A_1. There are two alternatives: A_3 is forced to come before A_2 (called *demotion*, see figure 2(b)) or A_3 is forced to come after A_1 (called *promotion*, see figure 2(c)).

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. Consider the situation shown in figure 3. In this case, the action A_3 interfere with the argument $\mathcal{B}$ because it negates a literal present in the argument. Note that $\overline{n}$ is an effect of A_3, where $n \in literals(\mathcal{B})$ (see definition 1). The argument step $\mathcal{B}$ was added to the plan to support the precondition b of the action step A_1. If A_3 make $\overline{n}$ 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.

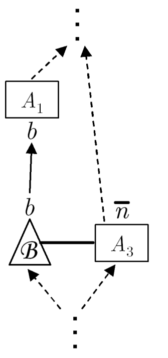

Figure 3: An action threaten and argument

Another thing to consider is that the existence of the argument $\mathcal{B}$ is not enough to have a warrant for b, because $\mathcal{B}$ could be defeated by a counterargument. This situation is shown in figure 4.

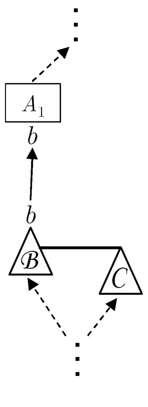

Figure 4: An argument is defeated

In this case the argument $\mathcal{B}$ is defeated by $\mathcal{C}$. Then, the literal b will not be warranted and the action A_3 will not be able to be executed.

4 CONCLUSIONS

We have analyzed the interaction between actions and arguments within partial order plans, in order to identify destructive interferences that can make a plan invalid. New types of interferences not present in classical Partial Order Planning have been found. We have to explore how this new interferences can be resolved and how the Partial Order Planning algorithm can be modified to introduce this new ideas.

APPENDIX A: ACTIONS AND DEFEASIBLE ARGUMENTATION

In DeLP, a literal L is *warranted* from the agent's knowledge base if there exists a non-defeated *argument* $\mathcal{A}$ supporting L. An argument structure $\mathcal{A}$ for a literal L, denoted $\langle \mathcal{A}, L\rangle$, is a minimal and consistent set of defeasible rules that allows to infer L. In order to establish whether $\langle \mathcal{A}, L\rangle$ is a non-defeated argument, a dialectical analysis is performed by considering *counter-arguments* that could be *defeaters* for $\langle \mathcal{A}, L\rangle$.

Besides its knowledge base, an agent will have a set of actions Γ that it may use to change its world. Once an action has been applied, the effect of the action will change the set Ψ. The formal definitions that were introduced in [4] are recalled below.

Definition 4 [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:

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

Notice that the notation *not* $\{C_1, \ldots, C_k\}$ represents $\{not\ C_1, \ldots, not\ C_k\}$.

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 whose negations *must not* be warranted. In this way, the satisfaction of the preconditions could also depend on the fact that some information is unknown (*un-warranted*).

Definition 5 [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, defined as before, is applicable if every precondition P_i in P has a warrant built from (Ψ, Δ) and every constraint C_i in C fails to be warranted.

Definition 6 [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:

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

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.

In [4], we have shown that the interaction between actions and the defeasible argumentation formalism is twofold. On one hand, as stated by Definition 5, defeasible argumentation is used for testing preconditions and constraints through the warrant notion. On the other hand, actions may be used by agents in order to change the world (actually the set Ψ) and then have a warrant for a literal L that has no warrant from the current knowledge base (Ψ, Δ).

References

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

[2] Garcia, D. R., Simari, G. R., and Garcia, A. J. Combining Partial Order Planning with Defeasible Argumentation. In *Proceedings of the VII Workshop de Investigadores en Ciencias de la Computación (WICC 2005).* (May 2005), Universidad Nacional de Rio Cuarto, Provincia de Córdoba, Argentina., pp. 354–358.

[3] Penberthy, J., and Weld, D. S. 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).

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

[5] Simari, G. R., and García, A. J. Using defeasible argumentation in progression and regression planning: Some preliminary explorations. In *Proceedings of the VIII Congreso Argentino en Ciencias de la Computación* (Oct. 2002), Universidad de Buenos Aires, Argentina, pp. 273–283.

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

[7] Weld, D. S. Recent advances in AI planning. *AI Magazine 20*, 2 (1999), 93–123.