Combining Partial Order Planning with Defeasible Argumentation

Diego García drg@cs.uns.edu.ar
Guillermo Simari grs@cs.uns.edu.ar
Alejandro García ajg@cs.uns.edu.ar

Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
Tel: (0291) 459-5135 / Fax: (0291) 459-5136

This research line involves the study of combining defeasible argumentation and different planning techniques. In this work we focus on the use of argumentation in combination with Partial Order Planning.

1 Defeasible Reasoning and Actions

In previous work [SGC04, SG02, SG01], we have introduced a formalism where agents represent their knowledge about their environment using the language of Defeasible Logic Programming (DeLP) [GS04], and have a set of actions that they can execute in order to change the environment where they are performing their tasks. In this section 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*. The results already obtained for such argumentation-based form of logic programming will be used here freely, but a brief description of DeLP will be introduced below.

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 [SG01] are recalled below.

Definition 1 [**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:

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

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

Partially Supported by the Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096 - PICT 2003 15043) and SeCyT Universidad Nacional del Sur.

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 (*unwarranted*).

Definition 2 [**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 3 [**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.

In [SG01], we have shown that the interaction between actions and the defeasible argumentation formalism is twofold. On one hand, as stated by Definition 2, 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).

A simple formulation of a planning problem defines three inputs [Wel99]: 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.

2 Argumentation in Partial Order Planning

The formalism described above defines when actions are applicable and how to compute its effects, but it does not describe how to construct a plan for achieving agent's goals. The aim of the research line introduced here is to study the combination of this formalism with different planning techniques. In this work we focus on Partial Order Planning.

The basic idea behind a regression Partial Order Planning (POP) algorithm [PW92] 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 third step might possibly interfere with the precondition being supported by the 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 the argumentation formalism described above, 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. 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 definition is introduced for identifying this set of literals.

Definition 4 (Argument base) Given an argument structure $\langle B, h\rangle$ the argument base for $\mathcal{B}$ will be the set of all literals that appear in the bodies of the rules in $\mathcal{B}$, and are not a head of a rule in $\mathcal{B}$. For example, the set $\{d, e\}$ is the argument base for $\mathcal{B}$= { $(b \prec c, d)$, $(c \prec e)$ }.

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. We will analyze some of this issues through the following example. For the sake of simplicity we will use a propositional language and actions without constraints.

Example 1 Suppose that an agent has the following knowledge base: $\Psi = \{e, f, h\}$ and Δ= { $(b \prec c, d)$, $(c \prec e)$, $(d \prec i)$, $(\sim b \prec j)$ }. The agent's goal is $G = \{a, g\}$, and the available actions are:

$$\{a\} \xleftarrow{A_1} \{b\}, not\ \{\}$$

$$\{c\} \xleftarrow{A_2} \{e\}, not\ \{\}$$

$$\{d\} \xleftarrow{A_3} \{f\}, not\ \{\}$$

$$\{g, j\} \xleftarrow{A_4} \{h\}, not\ \{\}$$

Figure 1 show different (and possibly incomplete) plans obtained from choosing different alternatives to achieve the unsatisfied preconditions. The square nodes represent steps (actions). 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

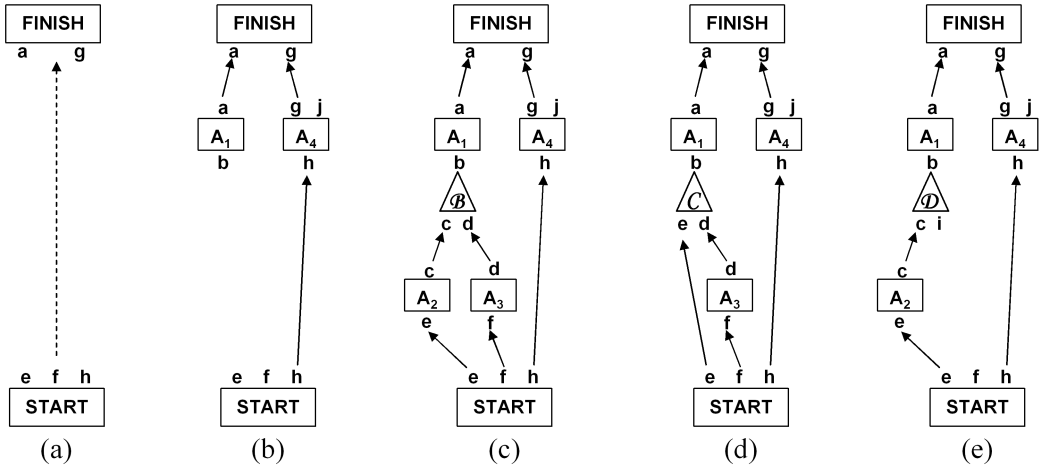

Figure 1: Different partial plans for Example 1

dashed arrow represents ordering constrains. Finally, triangles represents arguments. The literal at the top of the triangle is the literal supported by the argument, and the literals at the base of the triangle represent the argument base (Definition 4). Observe that Figure 1(a) depicts 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 argument base of $\mathcal{B}$ ($\{c, d\}$). 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 proper causal links are added.

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 argument 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.

Another thing to consider is that the existence of this argument $\mathcal{B}$ is not enough to have a warrant for b, because $\mathcal{B}$ could be defeated by a counter-argument. Again, this counter-argument will be also *potential* and it will depend on the decisions made later in the planning process. Therefore, as suggest by Pollock in [L.P98], the planning process should be done in an optimistic way. That is, plan for each subgoal separately, assuming that all the potential arguments constructed are non-defeated, until all preconditions are satisfied in some way. After that verify that the plan obtained do not contain destructive interferences: called threads in POP and defeaters in our approach.

For example, in the case depicted in Figure 1(c) the planning process has not been done following the traditional POP algorithm introduced above. Instead, it has been done in an optimistic way, without checking for "threads" as new steps or arguments were added to the plan. Therefore, once a possible plan is obtained, the planner must search for defeaters for the used arguments. If a defeater is found, then the planner will attempt to repair the plan reordering the plan steps. Following our example, if action A_4 is executed before A_1 then at the moment that b is needed for A_1 the argument $\mathcal{B}$ is defeated with $\{\sim b \prec j\}$ (because j is an effect of A_4). However, if the plan is reordered to execute A_1 before A_4, then at the state where $\mathcal{B}$ is used for supporting b, no defeater for $\mathcal{B}$ exists.

During the planning process the trade-off between using actions or arguments could affect the search space. Considering the agent's knowledge base of Example 1 there are two more potential arguments for supporting b: $\mathcal{C}$= $\{(b \prec c, d),\ (c \prec e)\}$ and $\mathcal{D}$= $\{(b \prec c, d),\ (d \prec i)\}$. For example, as shown in Figure 1(d), a shorter plan can be obtained using the argument $\mathcal{C}$ to support b, however, if the argument $\mathcal{D}$ is used (see Figure 1(e)) it is not possible to find a plan because the precondition i can not be satisfied.

References

[GS04] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[L.P98] John L.Pollock. Defeasible Planning. In *Integrating Planning, Scheduling, and Execution in Dynamic and Uncertain Environments AIPS Workshop.* Ralph Bergmann and Alexander Kott, Cochairs., 1998.

[PW92] 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.

[SG01] Guillermo R. Simari and Alejandro 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, October 2001. ISBN 987-96-288-6-1.

[SG02] Guillermo R. Simari and Alejandro J. García. Using defeasible argumentation in progression and regression planning: Some preliminary explorations. In *Proceedings of the VIII Congreso Argentino en Ciencias de la Computación*, pages 273–283. Universidad de Buenos Aires, Argentina, October 2002.

[SGC04] Guillermo R. Simari, Alejandro J. García, and Marcela 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.

[Wel99] Daniel S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.