1

Defeasible Argumentation-based Epistemic Planning with Preferences

Juan C. L. Teze, Lluis Godo, Gerardo I. Simari

ABSTRACT. Many real-world applications of intelligent systems involve solving planning problems of different nature, oftentimes in dynamic environments and having to deal with potentially contradictory information, leading to what is commonly known as epistemic planning. In this context, defeasible argumentation is a powerful tool that has been developed for over three decades as a practical mechanism that allows for flexible handling of preferences and explainable reasoning. In this chapter, we first motivate the need to develop argumentation-based epistemic planning frameworks that can be leveraged in real-world applications, describe the related literature, and then provide an overview of a recently-proposed approach to incorporate defeasible argumentation and preferences into automated planning processes. In particular, the framework incorporates conditional expressions to select and change priorities regarding information upon which plans are constructed. We describe its main properties, analyze its strengths and limitations using an illustrative use case, and discuss several future research directions that can be taken to further develop it.

1 Introduction

Planning is a research area within Artificial Intelligence (AI) that addresses the problem of obtaining a set of actions to achieve a specific goal given a description of the initial state of the world. Recently, the consideration of *epistemic* elements in building a plan has revealed a useful new perspective in the area: "*Epistemic planning is the enrichment of planning with epistemic notions, that is, knowledge and beliefs*" [**?**; **?**; **?**; **?**]. Various frameworks for planning have been proposed allowing for a formalisation and mechanization of knowledge-based reasoning in the planner itself. A central feature of classical frameworks is that their domain descriptions assume a fully observable, static, and deterministic world, which might lead to contradictions when the available knowledge is incomplete or inconsistent. In [**?**], the author concludes that since epistemic cognition is *defeasible*, a planning agent must be prepared to revise its plans as soon as its beliefs change, and may need to acquire more information through reasoning to solve a planning problem.

Defeasible argumentation is a form of reasoning about beliefs that can be used to exploit the contents of knowledge bases in the context of possible inconsistencies [**?**; **?**]. Specifically, the fundamental process in defeasible argumentation is to confront reasons to support or dismiss a conclusion that is under

scrutiny. An analysis mechanism supports this process by obtaining arguments for and against such conclusion, and then comparing those in conflict in order to reach a decision regarding acceptance.

Several works have proposed using argumentation to enhance planning systems. Particularly, planning problems have been primarily addressed from two points of view: Practical reasoning and Automated planning. In the context of the former, *i.e.,* reasoning about what to do next, a number of attempts have been made to leverage argumentation [**?**]. There are many ways to engage in practical reasoning, which make the task of formulating an argumentation-based planning system more complicated, mainly when investigating more specific aspects of rationality [**?**]. Using and instantiating Dung's argumentation framework [**?**] has been the predominant approach in practical reasoning – see, e.g. [**?**; **?**] – while another research line [**?**; **?**] closer to automated planning has explored how to use argumentation to guide the reasoning process. In general, in the latter approaches defeasible argumentation is used as the inference mechanism to reason about the preconditions and effects of actions in a planner system, especially in dynamic domains dealing with incomplete and contradictory information, which is often the case in real-world planning scenarios.

Actually, to solve a planning problem, a planning system with an appropriate set of actions should be provided. In classical planning, a general assumption on the representation of these actions is that it must encapsulate all the possible preconditions and effects that are relevant to solve the planning problem. Consider for example a scenario where a service robot agent has ordered some food from a restaurant and it is about to receive it at home by means of an action "receiving a food delivery service at home". A relevant effect can be "having a food delivery box at home". However, there could be other consequences that could be obtained but considered irrelevant, and thus not included in the representation of the action. For example, if the payment of the delivery service is done at the moment with a debit card, the customer will automatically have less money in her bank account. Therefore, instead of including all the possible effects in the representation of the actions, the system could be provided with a reasoning mechanism for obtaining those consequences that follow from the effects of an action. For example, "the ordered food is at home" could be considered as a *plausible* consequence of the effect that "there is a food delivery box at home". To do so, besides having the action specified in the planner, a possibility is to include extra knowledge in the form of a defeasible rule, such as "having a food delivery box at home is a reason to believe that the ordered food is at home".

It is important to remark that if "the ordered food is at home" is considered as an effect of the action, then it will be difficult to handle exceptions like "the delivered food is not what was ordered". However, this kind of problem can be properly handled by argumentation formalisms. For example, "there is a food delivery box at home but the delivered food is not what was ordered, is a

reason to believe that the ordered food is not at home". In particular, classical planning systems do not perform any type of reasoning over the effects of actions. In dynamic domains, it is a complex task to determine in advance what the effects of actions are because the information is constantly changing and depends on many factors. In this context, defeasible argumentation-based epistemic planners have been effectively applied in formalizing planning domains [**?**; **?**; **?**]; these approaches are characterized by the use of defeasible reasoning for the epistemic tasks performed over the represented knowledge.

Classical planning aims at finding a sequence of actions that, starting from an initial state, leads to a goal state. However, it is often the case that certain approaches are focused not only on the final goal state after plan execution, but also they attempt to address other important aspects, such as satisfying users' preferences [**?**], value-based selection of actions [**?**; **?**], or complying with norms imposed on the planner establishing what the system is required to do under certain conditions [**?**]. More specifically, modeling user preferences with explicitly-specified priorities has attracted the attention of many researchers. However, and despite its importance in the reasoning process, most of the existing argumentation-based planning systems do not provide additional capabilities for dynamically changing the preferences expressed by these priorities when a plan is being constructed.

In this chapter, we survey the main approaches in the literature concerning all the above-mentioned issues. However, the purpose of this survey is neither to cover the whole range of argumentation-based planning approaches nor to solve open questions or particular cases that have not been addressed so far; thus, we do not aim to have an exhaustive coverage of the subject. The rest of the chapter is structured in two main sections. In Section 2, we give an overview of different approaches in the area of planning with argumentation studied in the literature, while in Section 3 we present a description of a specific approach to deal with the handling of (contextual) preferences when a plan is formulated. In particular, we present the P-APOP algorithm proposed in [**?**], and summarize a set of computational complexity results. Finally, in Section 4 we offer our conclusions and discuss several challenges for research and hurdles that must be addressed on the path to obtaining fully working solutions.

2 Related Argumentation-based Planning Efforts

There are many challenging areas related to planning that have been addressed in the literature, and there is clearly much work still to be done in addressing epistemic planning issues. As a more general presentation of a set of epistemic planning-related research questions, we propose a list of aspects to be considered when trying to solve complex planning problems – presented in Table 1 – that motivate the criteria used to classify the different approaches presented in the next section.

This section reports a summary of approaches focusing on the use of argumentation in epistemic planning, and the handling of preferences. We will first

Research Questions
How can argumentation theory capture practical reasoning?
How can argumentation be exploited to guide the reasoning process, specifically for the selection and organization of actions?
How can argumentation theory be leveraged for plan search in cooperative scenarios?
How can the relationship between an agent's values and the construction of plans be formalized?
How can a set of agents achieve a goal jointly following a same plan?
What is the course of action to adopt in the presence of different goals and norms?
How can argumentation theory be exploited to explain the results of planning systems?
How can the notion of preference be embedded in argumentation and epistemic planning formalisms?
How can preferences be used to compute an optimal plan?
How can a planning system handle contextual preferences?

Table 1. A selection of the main research questions addressed by different lines of work in epistemic planning.

briefly touch upon some works that combine argumentation with planning, and then we focus particularly on works that incorporate the representation of and reasoning with preferences in the formalism.

2.1 Planning with Argumentation

In many real-world planning applications, it is common to encounter situations where unresolved contradictory and/or incomplete information occurs. Argumentation has become a very active research field because of its effective computational capacity to capture and solve conflicts, and there have been many research efforts towards the development of argumentation-based planning systems in the last two decades. In the following we discuss relevant contributions in different types of argumentation-based planning formalisms.

Practical Reasoning. A number of attempts have been made to address how argumentation theory can capture practical reasoning. Argumentation-based practical reasoning employs the conflict resolution capabilities of argumentation theory to solve conflicts between beliefs, intentions, and desires. Different approaches have dealt with these aspects; for instance, [**?**] introduces a formalism for agents following the BDI approach to reason about desires (generating desires and plans to achieve them). Argumentation-based proposals have also been used to compute the set of intentions to be pursued, or the resolution of incompatibilities among pursuable goals [**?**]. Motivated by the requirements of autonomic computing systems, [**?**] proposes the architecture of an Autonomous, Normative and Guidable agent (ANGLE) and its extended defeasible logic-based knowledge representation, including observations and motivational knowledge. In this formalism, the reasoning and decision-making tasks adopt argumentative deliberation based on dynamic theories. Other works, such as [**?**], follow the notion of argument schemes proposed by Walton [**?**]. Other approaches using argumentation in a normative environment were pro-

posed in [?; ?].

Automated Planning. Unlike argumentation-based approaches for practical reasoning, some planning formalisms have exploited the use of argumentation as a mechanism to guide the reasoning process, primarily concerned with the computational process for the selection and organization of actions. One of the well-known works on building a planner based on a defeasible reasoner was proposed in [?], in which Pollock presents OSCAR, an implemented architecture whose defeasible reasoner essentially performs a defeasible search for plans. In [?; ?], the authors introduce an argumentation-based formalism for constructing plans using partial order planning techniques, called DeLP-based partial order planning (DeLP-POP). In this approach, action preconditions can be satisfied either by actions' effects or conclusions supported by arguments, so actions and arguments are combined to construct plans. Actually, DeLP-POP is an extension of the POP algorithm that considers actions and arguments as planning steps and resolves the interferences that can appear. In [?; ?; ?], DeLP-POP is extended to multi-agent cooperative planning, while [?] presents a planning system based on DeLP to reason about context information during the construction of a plan – the system is designed to operate in cooperative multi-agent environments. Each step of the construction of a plan is discussed among agents following a proposed dialogue mechanism that allows agents to exchange arguments about the conditions that might affect an action's feasibility according to their distributed knowledge and beliefs. Temporal defeasible reasoning has also been studied in the planning literature, where for instance Pardo and Godo [?; ?] presented a distributed multiagent planning system for cooperative tasks. The main feature of the proposal is the development of a planning approach based on t-DeLP, an extension of DeLP for defeasible temporal reasoning. The authors also propose a dialogue-based algorithm for plan search in cooperative scenarios.

The work of [?] concerns epistemic planning problems, focusing on an argumentation-based approach. The paper discusses first steps in developing an approach to handle contextual preferences that can dynamically change based on knowledge-based priorities. They introduce a generic architecture, independent of the underlying formalism and reasoning mechanisms, as well as a set of guidelines to support knowledge and software engineers in the analysis and design of planning systems leveraging this preference handling capacity. The authors also present a concrete instantiation based on Possibilistic Defeasible Logic Programming [?]. Recently, [?] presents a revised, refined, and extended version of [?], where the main criteria employed to decide which actions to keep during the construction of plans, by using contextual conditional-preference expressions associated with each action, are formally defined. It also discusses possible interferences that can appear when such expressions are used, and it presents an extension of the APOP algorithm [?] for this setting with contextual preferences.

Following the idea of value-based argumentation [?], there have also been

approaches that focused on providing grounds for formalizing the relationship between values and actions, and integrating defeasible argumentation into the agent reasoning process. For instance, in [**?**] the values that an agent holds are used to compare plans, and several comparison strategies are formally defined.

Planning Problems in Multiagent Environments. In multiagent environments, agents may need to jointly follow a course of action in order to achieve a goal. The different viewpoints that agents have on the environment may cause disagreements, and reaching an agreement requires the alignment of viewpoints. Argumentation provides natural ways for conflict resolution in collaborative decision making. Many works have advanced the state of the art in argumentation-based multiagent planning. In [**?**; **?**] the authors investigate the use of argumentation to solve conflicts between planning proposals caused by inconsistency between beliefs. Another interesting work that combines the benefits of argumentation in multiagent environments emphasizes the use of defeasible temporal reasoning for negotiation dialogues [**?**; **?**]; an extended and revised version of this work is proposed in [**?**]. In the same vein, [**?**] introduces a proposal that models the argumentation process as a planning process, and obtains an argumentation-based negotiation plan. In [**?**; **?**], the authors present a multiagent extension of the DeLP-based Partial Order Planning (POP) framework [**?**] for cooperative planning. Apart from individual goals, the system may require to follow societal norms that promote systems that follow the right behavior. The work of [**?**] addresses the question of what is the best course of action to adopt in the presence of different goals and norms, proposing a solution based on argumentation schemes for deliberative dialogues in multiagent environments.

Explainable Planning. Explainable AI Planning [**?**] is a fairly recent research area that involves explaining the outcome and results of planning systems. The relevant question here is how can argumentation theory be exploited to explain the results of planning systems. Argumentation has been widely recognized by the Explainable AI community [**?**] as a powerful logical model of reasoning that is capable of explaining the behavior of a system by linking any system decision to the evidence supporting it. Some recent approaches like [**?**] build around a set of argument schemes that create arguments that give explanations for a plan and its key elements (*i.e.,* actions, states, and goals). In [**?**], the explanations of justifiability of the best plan are generated using an argumentation-based dialogue. A proposal for resolving planning problems with assumption-based argumentation (ABA) was presented in [**?**]. This work proposes to generate explanations for both planning solutions as well as failed plans extracted from *dispute trees* [**?**]. The work of [**?**] presents a prototype system implementation based on ASPIC [**?**] for building arguments that justify why a plan should be executed. Two alternatives for plan explanation are considered: visual plan explanation via graphical representations, and textual representation of a plan in a natural language created through a dialogue-based approach, where participants take turns to make utterances that are used to establish whether

some argument (and therefore its conclusions) is justified.

2.2 Representation and Reasoning with Preferences

In many planning approaches [**?**; **?**; **?**], modeling user preferences with explicitly-specified priorities plays a significant role, especially in decision-making processes. This priority information is beneficial in the selection of appropriate knowledge, and guides the planning process according to user needs. In this section, we discuss how the notion of preference has been embedded in argumentation and epistemic planning formalisms.

2.2.1 Preferences in Defeasible Argumentation

Defeasible Argumentation formalisms have received increased attention as an advanced mechanism to formalize essential parts of what is known as commonsense reasoning. One of the main issues the argumentative reasoning process must address is confronting reasons to support or dismiss a claim that is under scrutiny. For this purpose, there is a need for an analytical mechanism that follows well-understood steps, starting by obtaining arguments and then comparing those in conflict to determine which arguments prevail; this last step requires a comparison, which in turn needs a preference criterion on the set of arguments to evaluate the strength or importance of arguments in order to reach a decision.

Despite the clear significance of the outcome of a comparison among arguments, there is neither a unique way of establishing a preference relation between arguments nor a consensus in the argumentation literature regarding which criterion should be used; for a comprehensive overview, see for instance [**?**]. For example, some approaches choose to use a criterion that considers an explicit order over rules [**?**; **?**], whereas others consider an order over literals [**?**; **?**; **?**] or even social values [**?**]. In [**?**], the authors extend the work of [**?**] in order to take into account multiple values and various kinds of preferences over values. In [**?**], the preference is defined in terms of the strength or credibility of those agents that contribute with pieces of information to the argument. Based on the idea of prioritized norms, [**?**] shows different variants of lifting priorities over norms to priorities on arguments themselves, allowing to capture a preference order over arguments. Differently from [**?**], where an extension of ASPIC$^+$ is proposed to use preferences to resolve attacks, [**?**] introduces ABA$^+$ considering preferences on assumptions rather than (defeasible) rules. In [**?**], the author presents a formalism in Dung's abstract argumentation framework for metalevel argumentation-based reasoning about preferences between arguments, and applies it to Prakken and Sartor's argument based logic programming with defeasible priorities (ALP-DP).

Other approaches, such as [**?**; **?**; **?**], define a criterion based on a generalized specificity principle. Furthermore, there exist other formalisms that use a combination of several fixed and predefined criteria [**?**; **?**; **?**; **?**], while others simply consider a general preference relation [**?**; **?**]. Usually, in current argumentation formalisms, the definition of the argument comparison criterion is

either fixed and embedded in the system, or it is modular. See the works of [**?**; **?**; **?**] for reviews of argument preference criteria present in the argumentation literature.

2.2.2 Preferences in Epistemic Planning

In the work of [**?**], the authors argue that users' preferences are of great importance in selecting a plan for execution when the space of solution plans is dense. While there is a significant body of research on preference within classical planning theory [**?**; **?**; **?**; **?**; **?**], most of the research in epistemic planning has particularly been focused on methodologies and issues related to computational efficiency. Relatively limited efforts have been dedicated to addressing other important aspects, such as generating high quality plans satisfying users' preferences and constraints. For example, [**?**] presented a first proposal of a language for specification of preferences for planning problems and included a logic programming encoding of the language based on Answer Set Programming (ASP). The language allows to handle four different preference categories: about a state, an action, a trajectory, or multi-dimensional preferences. Recently, [**?**] proposed an automated planning approach for the task of planning with epistemic preferences, which incorporates weighted preferences and computes the optimal plan by maximizing the sum of weights of the preferences satisfied. In [**?**], the authors propose the use of a preference-based planning algorithm that represents the argument selection criterion into the agent's mental state as preferences. The algorithm uses the agent's preferences in order to select the best actions.

Regarding approaches using argumentation, [**?**] focused on providing grounds for formalizing the relationship between values and actions, and for integrating defeasible argumentation into the agent reasoning process. In this formalism, the main idea is that of using values to compare plans, and several comparison strategies are formally defined. The authors propose to arrange values hierarchically, and exploit an agent's preferences over values using such a hierarchy. Finally, as already mentioned above, [**?**] and its extension [**?**] deal with defeasible argumentation-based epistemic planning with an approach to handle contextual preferences that can dynamically change via knowledge-based priorities.

The research efforts in the area discussed in this section are summarized in Figure 1. As we have mentioned, the formalization and use of mechanisms for handling preferences have not been widely adopted in the epistemic planning literature. In fact, tackling the fundamental question of whether the notion of preference can be embedded in argumentation and epistemic planning formalisms makes it a particularly challenging research topic. In this context, in the next section we provide some details of the approach developed in [**?**], which takes that direction.

Reference	**PR**	**AP**	**ME**	**EP**	**P**
Pollock *et al.*[**?**]		✓			
Rahwan *et al.*[**?**]	✓				
García *et al.*[**?**]		✓			
Bench-Capon *et al.*[**?**]	✓				
Belesiotis *et al.*[**?**]		✓	✓		
Amgoud *et al.*[**?**]	✓				
Monteserin *et al.*[**?**]		✓	✓		✓
Pardo *et al.*[**?**]		✓	✓		
Toniolo *et al.*[**?**]	✓		✓		
Shams *et al.*[**?**]	✓		✓	✓	
Fan *et al.*[**?**]		✓		✓	
Pardo *et al.*[**?**]		✓	✓		
Teze *et al.*[**?**]		✓			✓
Oren *et al.*[**?**]		✓		✓	
Shams *et al.*[**?**]	✓				
Teze *et al.*[**?**]		✓			✓
Teze *et al.*[**?**]		✓			✓
Parsons *et al.*[**?**]		✓		✓	

Figure 1. Comparison of argumentation-based planning approaches in terms of five categories, where check marks indicate a focus on the respective category. Abbreviations: **PR** (*Practical Reasoning*), **AP** (*Automated Planning*), **ME** (*Multiagent Environment*), **E** (*Explainability*), and **P** (*Preferences*).

3 Argumentation-based Epistemic Planning with Preferences

With the goal of providing more details about a specific approach, in this section we present an overview of an epistemic planning framework proposed in [**?**; **?**], which is a formalism that incorporates defeasible argumentation as a reasoning mechanism in the construction of plans. The main novelty of this epistemic planning formalism centers on introducing a way to select the priority assignment mechanism to modify the preferences among different pieces of defeasible knowledge as the planner reasons and chooses which actions to add to a plan. In the following, we start by describing a general architecture for defeasible argumentation-based epistemic planning, and then show a P-DeLP-based particular instantiation.

Figure 2. The Epistemic Planning Framework based on Argumentation (figure reproduced from [**?**]).

3.1 An Argumentation-based Epistemic Planning Framework

The development of planning systems with defeasible reasoning and preferences can be a very complex task involving several stages toward obtaining the final system. In [**?**], instead of a specific solution, a set of guidelines is introduced to support knowledge and software engineers in the analysis and design of planning systems, focused on five central stages:

1. *Planning domain analysis*: In complex and dynamic environments, planning systems eventually may require dealing with contradictory and incomplete knowledge about the domain. In this context, structured argumentation has played a crucial role in capturing and representing this type of knowledge. This stage is aimed at providing a detailed and precise description of the planning domain and the user's preferences, which includes: a knowledge base in a formal language expressing domain information, and a specification of a preference relation over pieces of knowledge. These preferences reflect the importance or priority of the information that arguments built in the reasoning process will contain.

2. *Planning problem analysis*: In planning, the classic problem involves finding a sequence of actions that, starting from an initial state, leads to a goal state. A precise description of the planning problem requires a deeper analysis to identify its properties. Several dimensions need to be considered, such as whether multiple actions can be taken concurrently or if only one action is possible at a time, whether the objective is to reach a designated goal state or to maximize a reward function, the presence of one agent or multiple agents, or whether actions have associated probabilities. These issues should be carefully analyzed during this stage. Also, as we have already mentioned, it is often the case that certain planning approaches are concerned not only with the final goal state after plan execution, but also with attempting to address other important aspects, such as users' preferences [**?**] or value-driven actions [**?**].

3. *Reasoning mechanism*: This mechanism is in charge of interpreting the available domain knowledge, generating arguments, and then comparing those in conflict to decide on acceptance. A reasoner contains three main components: an *Inference Mechanism*, which carries out inferences based on available knowledge to be used in the construction of plans; a *Conflict Solver*, which establishes a preference relation over the set of arguments through an argument comparison criterion; and a *Semantic Analyzer*, which aims at determining the acceptability of arguments by considering

the interaction between them. This last process can be done declaratively via conditions that a set of acceptable arguments must meet [**?**], or procedurally with a specific algorithm [**?**].

4. *Planning mechanism*: Responsible for the general algorithm driving the main planning system functionality, which consists in coordinating the interactions among the components mentioned above, and obtaining a sequence of actions to achieve the desired goals making use of defeasible reasoning in the process. Most of the proposals in the literature generally consider one of the following two approaches: either the whole plan is viewed as an argument, and then defeasible reasoning is performed over complete plans, or it is used as a tool for determining which actions are applicable in a given state. Planning algorithms are also mainly based on two approaches: *progression* planning and *regression* planning. The former searches forward from a given initial state until a goal state is reached, while the latter tries to improve this situation by beginning from the goal state and generating the plan in inverse order.

5. *Output design*: This step includes effective ways to interpret the process by which the planning decisions are made. Plan explainability is essential for helping users to understand and improve trust in plans [**?**], and such explanations can take on several forms. Visual plan explanation [**?**] presents a graphical view of a plan, with nodes representing actions, edges linking them, and different filtering options available to the user. Other approaches [**?**] involve a textual description of the plan in natural language. Related works [**?**] use argumentation for providing mechanisms to construct arguments that can be useful to justify why a plan should be executed.

Figure 2 schematically illustrates an epistemic planning framework based on the methodological guidelines described above.

In the rest of the section, we will sketch a particular instantiation of the above generic epistemic planning framework presented in [**?**]. We first recall the P-DeLP argumentation system upon which the planning formalism is built, then we present the planning formalism itself along with related algorithms, and finally we proceed to show computational complexity results associated with this framework.

3.2 P-DeLP: An Extension of the DeLP Argumentation Framework Dealing with Ordinal Preferences

Possibilistic Logic (see *e.g.,* [**?**] for full details) is a logic of qualitative uncertainty, alternative to other more numerical uncertainty models like the probabilistic one, where what really matters is the likelihood order induced on propositions by the uncertainty values they take, and not the absolute values themselves. It is thus an *ordinal* model that is very suitable for handling preferences [**?**; **?**; **?**]. Possibilistic Defeasible Logic Programming (P-DeLP) [**?**;

?] is a structured argumentation framework that extends the DeLP framework [**?**] by allowing to attach weights to argument conclusions. The ultimate answer to queries is based on the existence of warranted arguments computed through a qualitative dialectical analysis. The top-down proof procedure of P-DeLP is based on the one used in Defeasible Logic Programming.

In P-DeLP, a knowledge base represents domain knowledge and user preferences encoded as prioritized DeLP rules. Given a set of literals $\mathbf{L}$, a *weighted* clause is a pair $(R;\omega)$, where R is a rule $L \leftarrow L_1, \ldots, L_k$ or a fact L (*i.e.,* a rule with empty antecedent), $L, L_1, \ldots, L_k \in \mathbf{L}$, and the weight $\omega \in [0, 1]$ expresses the priority or preference degree of the clause, interpreted as a lower bound for the conditional necessity degree $Nec(L \mid L_1 \wedge \ldots \wedge L_k)$ in the case $R = L \leftarrow L_1, \ldots, L_k$, or a lower bound for the necessity degree $Nec(L)$ in the case $R = L$. Note that, by considering $Nec(L \mid L_1 \wedge \ldots \wedge L_k)$ we are following the usual notational conventions in Logic Programming [**?**] that regards the set of literals in the body of a clause $L_1, \ldots, L_k$ as a conjunction of these literals. Also, following [**?**], P-DeLP rules can be represented as schematic rules with variables; as usual in Logic Programming, schematic variables are denoted with initial uppercase letters. To keep the usual terminology in defeasible reasoning, we distinguish between *strict* and *defeasible* clauses: a clause $(R;\omega)$ is referred to as strict if $\omega = 1$ (top priority) and defeasible otherwise (*i.e.,* if $\omega < 1$). The higher the weight ω, the higher the priority of the clause. Given a set $\mathbb{P}$ of weighted clauses, often referred to as a P-DeLP program or simply a *program*, we will distinguish the set of all the clauses in $\mathbb{P}$ considered as strict, denoted Π, and the set of all the defeasible clauses in $\mathbb{P}$, denoted Δ. When useful, we will write $\mathbb{P} = (\Pi, \Delta)$ to refer to the set of weighted clauses, discriminating strict and defeasible clauses.

Example 1 *The following application domain was introduced in [**?**], and consists of a scenario where a cooking service robot was designed to prepare a meal considering the user's particular preferences. Consider the following P-DeLP program modeling the robot's knowledge.*

$$\Pi_1 = \left\{ \begin{array}{l} (open\_now(superfour);\ 1) \\ (\sim good\_products(superfour);\ 1) \end{array} \right\}$$

$$\Delta_1 = \left\{ \begin{array}{l} (suggest(S) \leftarrow open\_now(S);\ 0.2) \\ (\sim suggest(S) \leftarrow \sim good\_products(S);\ 0.7) \end{array} \right\}$$

Observe that the set Π_1 of strict clauses has two facts, and the set Δ_1 has two defeasible rules, which can be interpreted as follows: "S is a supermarket that is open now ($open\_now(S)$)" is a reason to suggest it, whereas if "S does not offer good products ($\sim good\_products(S)$)" then there exist reasons against suggesting it. Moreover, the weights attached to the defeasible rules indicate that the second rule has more priority or is more preferred than the first rule. Recall that weights in P-DeLP are purely ordinal, so what really matters here is the preference ordering they induce.

We will use the symbol "$\mathrel{|\!\sim}$" to denote the possibilistic inference meta-relation between a program $\mathbb{P}$ and a weighted literal $(L;\omega)$, *i.e.,* $\mathbb{P} \mathrel{|\!\sim} (L;\omega)$ will express that from $\mathbb{P}$ it is possible to build a sequence $(L_1;\omega_1),\ldots,(L_n;\omega_n)$ of weighted literals such that (a) $(L_n;\omega_n) = (L;\omega)$, and (b) each $(L_i;\omega_i)$ with $i < n$ either belongs to $\mathbb{P}$ or has been obtained by the application of the following *generalized modus ponens rule*

$$\frac{(H \leftarrow H_1,\ldots,H_k;\beta)\quad (H_1;\gamma_1),\ldots,(H_k;\gamma_k)}{(H;\min(\beta,\gamma_1,\ldots,\gamma_k))} \qquad [GMP]$$

where $(H \leftarrow H_1,\ldots,H_k;\beta) \in \mathbb{P}$ and all weighted literals $(H_1;\gamma_1),...,(H_k;\gamma_k)$ appear before $(L_i;\omega_i)$ in the sequence. Note that this rule is sound with respect to the semantics of necessity degrees as introduced for instance in [**?**]. Indeed, if $Nec(H_i) \geq \gamma_i$ for $i = 1,\ldots,k$ and $Nec(L \mid H_1 \wedge \ldots \wedge H_k) \geq \beta$, then $Nec(H) \geq \min(\beta,\gamma_1,\ldots,\gamma_k)$.[1]

A P-DeLP program $\mathbb{P} = (\Pi,\Delta)$ is said to be *contradictory* if, for some atom a, $\mathbb{P} \mathrel{|\!\sim} (a;\omega)$ and $\mathbb{P} \mathrel{|\!\sim} (\sim a;\beta)$, with $\omega > 0$ and $\beta > 0$. Since the strict part Π represents non-defeasible information, we will assume that Π is non-contradictory itself. When reasoning from a contradictory program $\mathbb{P}$, the P-DeLP system builds arguments from $\mathbb{P}$.

An *argument* for a literal L with necessity degree $\omega > 0$, denoted $\langle \mathcal{A},(L;\omega)\rangle$, is a minimal, non contradictory set of defeasible rules $\mathcal{A}$ such that together with the program's strict knowledge allows the derivation of L with a given weight ω (the smallest weight of the clauses involved in the derivation), that will be regarded as the conclusion supported by the argument $\mathcal{A}$.

Example 2 *The following arguments* $\langle \mathcal{A}_1,(suggest(superfour);0.2)\rangle$ *and* $\langle \mathcal{A}_2,(\sim suggest(superfour);0.7)\rangle$ *can be built from the P-DeLP program* $\mathcal{P}_1$ *presented in Example 1, where:*

$$\mathcal{A}_1 = \left\{ \; (suggest(superfour) \leftarrow open\_now(superfour);\ 0.2) \; \right\}$$

$$\mathcal{A}_2 = \left\{ \; (\sim suggest(superfour) \leftarrow \sim good\_products(superfour);\ 0.7) \; \right\}$$

and the literals $(suggest(superfour);\ 0.2)$ *and* $(\sim suggest(superfour);\ 0.7)$ *are obtained by applying the GMP inference rule presented above to the strict facts in* Π_1 *and the weighted rules in* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*, respectively.*

Given a program program $\mathbb{P}$ and a literal L as input, the answer of the P-DeLP system to the query L is based on checking for the existence of warranted arguments for L, computed through an exhaustive dialectical analysis

[1] Recall that a necessity measure Nec on a propositional language $\mathcal{L}$ is a mapping $Nec : \mathcal{L} \to [0,1]$ such that $Nec(\top) = 1$, $Nec(\bot) = 0$, and $Nec(\varphi \wedge \psi) = \min(Nec(\varphi),Nec(\psi))$. Then, the corresponding (qualitative) notion of conditional necessity is usually defined as follows: $Nec(\varphi \mid \psi) = Nec(\neg\psi \vee \varphi)$ if $Nec(\neg\psi \vee \varphi) > Nec(\neg\psi)$, and $Nec(\varphi \mid \psi) = 0$ otherwise.

that involves the construction and evaluation of arguments that either support or interfere with the query under analysis. That is, the warrant process evaluates whether there exists for some weight $\alpha > 0$ an argument $\langle \mathcal{A}, (L; \alpha) \rangle$ from $\mathbb{P}$ that cannot be defeated; see [**?**; **?**] for more details about the entire warrant process.

3.3 A P-DeLP-based Planning Framework Instantiation

Having prioritized information can particularly be useful to guide the reasoning process in a planning problem. One of goals in [**?**] was to allow the adjustment of the priority weights on rules to be used by P-DeLP's inference mechanism when selecting actions in the planning process. The priority degree associated with a defeasible rule is then context-dependent, where the notion of context is understood – in a general sense – as conditions favoring a particular priority criterion.

Given a finite set of rules $\mathbb{R}$, a *priority criterion* prc is formally defined as an assignment $\rho_{\mathsf{prc}} : \mathbb{R} \to [0, 1)$ of priority degrees to the rules in $\mathbb{R}$. For simplicity, we will write $\mathsf{prc}(R)$ instead of $\rho_{\mathsf{prc}}(R)$, and given a set of (weighted) defeasible rules Δ and a priority criterion prc, we will write Δ_{prc} to denote the set of rules resulting from updating the weights of the rules of Δ. If prc assigns the minimal weight 0 to a defeasible rule R (*i.e.,* if $\mathsf{prc}(R) = 0$), this means that R plays no role at all under criterion prc. On the other hand, note that by definition it is not allowed to assign a maximal weight 1 to a (defeasible) rule since in that case it would become a strict rule.

Example 3 *Consider the defeasible rules of the P-DeLP program $\mathcal{P}_1$ from Example 1, and the two criteria* $\mathsf{pref\_rocio}$ *and* $\mathsf{pref\_aldo}$ *prioritizing rules according respectively to the preferences of Rocio and Aldo, who are homeowners. The following are the corresponding sets of updated rules according to these criteria:*

$$\Delta_{\mathsf{pref\_rocio}} = \left\{ \begin{array}{l} (suggest(S) \leftarrow open\_now(S);\ 0.6) \\ (\sim suggest(S) \leftarrow \sim good\_produts(S);\ 0.4) \end{array} \right\}$$

$$\Delta_{\mathsf{pref\_aldo}} = \left\{ \begin{array}{l} (suggest(S) \leftarrow open\_now(S);\ 0.2) \\ (\sim suggest(S) \leftarrow \sim good\_produts(S);\ 0.9) \end{array} \right\}$$

Since arguments rely on defeasible knowledge, when they are evaluated it can be the case that there exist arguments supporting contradictory literals, so that a particular argument comparison strategy to deal with the conflicting arguments is required. A specific strategy presented in [**?**] relies on comparing the weights of arguments. Using this strategy and the priorities of Example 3 specified in $\Delta_{\mathsf{pref\_rocio}}$, the argument $\langle \mathcal{A}_1, (suggest(superfour); 0.6) \rangle$ is preferred over the argument $\langle \mathcal{A}_2, (\sim suggest(superfour); 0.4) \rangle$, since $\mathcal{A}_1$ provides a greater weight for the conclusion than $\mathcal{A}_2$.

One of the features of the planning framework we describe in this section is a

mechanism that dynamically modifies preferences (or priorities) among pieces of defeasible knowledge depending on the current state of the world the planner is acting upon, that will be described below. In the planning system, a *state of the world* Ψ is represented as a consistent set of literals, considered to hold true.

Example 4 *The following consistent set of facts can represent a possible state of the world in a given moment:*

$$\Psi_4 = \left\{ \begin{array}{l} lunchtime \\ open\_now(superfour) \\ superM(superfour) \\ \sim good\_products(superfour) \\ recipe(pastaPuttanesca) \end{array} \right\},$$

where it is lunch time, superfour is a supermarket that is open now but does not offer good products, and that a recipe for preparing pasta puttanesca is available.

Defeasible argumentation is used for reasoning over the preconditions to execute actions. Indeed, a set of domain defeasible rules Δ together with a set Ψ of literals describing the current state of the world define a P-DeLP program (Ψ^*, Δ), where $\Psi^* = \{(L, 1) \mid L \in \Psi\}$, upon which the planner system can perform defeasible reasoning about whether preconditions of a given action are warranted. We will denote by $\mathsf{warrL}(\Psi, \Delta)$ the set of literals warranted by the program (Ψ^*, Δ).

To do so in a specific context, the planning system can use a particular priority order over the defeasible knowledge that will be obtained after evaluating a given expression. The idea is to associate to every action a suitable conditional expression that will select, by means of *guards*, the priority criteria to be used in each given context depending on the world's current state.

A *guard* is a set of literals γ, and it is satisfied by a state Ψ when $\gamma \subseteq \Psi$. In its simplest form, a *conditional-preference expression* E can be just a priority criterion prc, and in that case the priority assignment corresponding to this criterion is applied over defeasible rules. In general, E can be of the form $E = [\gamma : E_1; E_2]$, where γ is a guard and where E_1 and E_2 can be in turn either priority criteria or further conditional expressions. In such a case, if E is evaluated, and γ is satisfied in the current state Ψ (*i.e.,* $\gamma \subseteq \Psi$), then E_1 is evaluated; otherwise, E_2 is evaluated. This recursive evaluation procedure is applied until a priority criterion is obtained.

Example 5 *Let us consider the following conditional preference about the two priority criteria introduced in Example 3:*

- *"If it is lunch time, use Rocio's preferences, otherwise use Aldo's preferences",*

This informal statement can be captured with the following conditional-preference expression:

$$E_1 = [\{lunchtime\} : E_2; E_3],$$

where the (non-conditional) expressions E_2 and E_3 stand for

$$E_2 = \mathsf{pref\_rocio},$$

$$E_3 = \mathsf{pref\_aldo}.$$

Consider now the state Ψ_4 introduced in Example 4. It is clear that the guard "lunchtime" is satisfied by the state Ψ_4, and thus the result of evaluating E_1 at Ψ_4 is the priority criterion $E_2 = \mathsf{pref\_rocio}$.

Having defined what conditional-preference expressions are, they can be used to formally define the set of actions a planner system may use to change the world and achieve its goals. Three elements specify an action A: its preconditions P, its consequences X, and the preferences E under which P will be evaluated.

An *action* is a triple $\mathsf{A} = \langle \mathsf{X}, \mathsf{P}, E \rangle$, where $\mathsf{X} = \{X_1, X_2, \ldots, X_n\}$ is a consistent set of literals representing the consequences of executing A, $\mathsf{P} = \{P_1, P_2, \ldots, P_n\}$ is a set of literals representing the preconditions that need to be satisfied before A can be executed, and E is a conditional-preference expression representing the preferences under which to evaluate preconditions P. We will use the following notation for actions:

$$\{X_1, X_2, \ldots, X_n\} \xleftarrow{(\mathsf{A}, E)} \{P_1, P_2, \ldots, P_n\}.$$

Intuitively, given a context represented by a state Ψ and a defeasible knowledge base Δ, an action $\mathsf{A} = \langle \mathsf{X}, \mathsf{P}, E \rangle$ specifies that "if all preconditions of A are warranted by the argumentation system $(\Psi, \Delta_{\mathsf{prc}})$, where prc is the criterion obtained by evaluating E at Ψ, then after executing A the postconditions X will be added to the state Ψ".

Example 6 *Consider the application domain presented in Example 1, and the conditional-preference expressions of Example 5. The actions that the robot can perform are the following:*

$$\mathbf{A}_6 = \left\{ \begin{array}{l} \{food\_prod\_ordering\} \xleftarrow{(\mathsf{order\_food\_products}, E_1)} \{recipe(R), superM(S), suggest(S)\} \\ \{ing\_ready\} \xleftarrow{(\mathsf{search\_storage}, E_1)} \{recipe(R), storage(R)\} \\ \{ing\_ready\} \xleftarrow{(\mathsf{receive\_food\_products}, E_2)} \{food\_prod\_ordering\} \\ \{homemade\_meal\} \xleftarrow{(\mathsf{cooking}, E_1)} \{ing\_ready\} \end{array} \right\}$$

These actions can be interpreted as follows:

— $\mathsf{order\_food\_products}$*: ordering food products from a supermarket. There must exist a supermarket available for making an order.*

— search_storage*: searching for the correct ingredients from the house storage. The ingredients in the recipe must be in the storage.*

— receive_food_products*: receiving the supermarket's products at home. There must exist a food product order.*

— cooking*: cooking at home. All of the recipe's ingredients must be available.*

Apart from the domain knowledge to reason during the planning process, the planner will have a set of actions that will be available for modifying the world.

Formally, a *preference-based planning domain* is a triple $(\Delta, \mathbf{C}, \mathbf{A})$ where:

- Δ is a set of defeasible rules.
- $\mathbf{C}$ is a set of priority criteria over rules of Δ.
- $\mathbf{A} = \{\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n\}$ is a set of actions, where for each $\mathsf{A} \in \mathbf{A}$, and $\mathsf{A} = \langle \mathsf{X}, \mathsf{P}, E \rangle$, such that for every prc in E it holds that $\mathsf{prc} \in \mathbf{C}$.

As already mentioned, checking whether an action can be executed involves checking its applicability, *i.e.,* checking whether the literals of the set of preconditions can be warranted. After an applicable action is executed, the state itself is consistently modified with each effect after removing any possible conflict. The new state resulting from executing an action A in the state Ψ will be denoted by $\Psi^{\mathsf{A}} = (\Psi \setminus \overline{\mathsf{X}}) \cup \mathsf{X}$, where $\overline{\mathsf{X}}$ is the set of the complemented literals in X.

Example 7 *Consider the set of defeasible rules Δ_1 defined in Example 1, the set of criteria $\mathbf{C}_3 = \{\mathsf{pref\_rocio}, \mathsf{pref\_aldo}\}$ of Example 3, and the set of actions $\mathbf{A}_6$ presented in Example 6. Suppose the robot's planning system has the following domain $(\Delta_1, \mathbf{C}_3, \mathbf{A}_6)$ and the state Ψ_4 presented in Example 4, where:*

$$\Psi_4 = \left\{ \begin{array}{l} lunchtime \\ open\_now(superfour) \\ superM(superfour) \\ \sim good\_products(superfour) \\ recipe(pastaPuttanesca) \end{array} \right\}.$$

Consider now the action order_food_products *in $\mathbf{A}_6$ and the priority criterion* pref_rocio *obtained after evaluating E_1. This action is applicable in Ψ_4 according to the priorities defined by* pref_rocio *because one of its preconditions, $recipe(pastaPuttanesca)$, is in Ψ_4 and its other precondition, $suggest(superfour)$, belongs to* $\mathsf{warrL}(\Psi_4, \Delta_{\mathsf{pref\_rocio}})$ *since there exists a non-defeated argument for $\langle \mathcal{A}_1, (suggest(superfour); 0.6) \rangle$ where:*

$$\mathcal{A}_1 = \left\{ \; (suggest(superfour) \leftarrow open\_now(superfour);\ 0.6) \; \right\}$$

The resulting state of executing the action order_food_products *in state* Ψ_4 *is then the following:*

$$\Psi^{\mathsf{sel\_recipe}(pastaPuttanesca)} = (\Psi_4 \setminus \overline{\mathsf{X}}) \cup \mathsf{X} = \left\{ \begin{array}{l} lunchtime \\ open\_now(superfour) \\ superM(superfour) \\ \sim good\_productse(superfour) \\ recipe(pastaPuttanesca) \\ food\_prod\_ordering \end{array} \right\}$$

where $\mathsf{X} = \{food\_prod\_ordering\}$.

Since the execution of an applicable action leads to a new state, another action could be applicable at this new state, and so on. A sequence $\mathsf{S} = [\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ will be regarded as an *applicable sequence* of actions at a state Ψ if (1) A_1 is applicable at Ψ, and (2) every action A_i, $2 \leq i \leq n$, is applicable in $(\cdots(\Psi^{\mathsf{A}_1})\cdots)^{\mathsf{A}_{i-1}}$. We will use Ψ^{S} or $\Psi^{[\mathsf{A}_1,\ldots,\mathsf{A}_n]}$ as a shorthand for $(\cdots(\Psi^{\mathsf{A}_1})\cdots)^{\mathsf{A}_n}$. In fact, the main aim of any planning system is to find a sequence of actions that, starting from an initial state, leads to a state where a given goal is satisfied.

A *preference-based planning problem* is a tuple $(\Psi, \Delta, \mathbf{C}, \mathbf{A}, \mathbb{G})$, where:

— Ψ is a consistent finite set of weighted literals representing an initial state,

— $(\Delta, \mathbf{C}, \mathbf{A})$ is a preference-based planning domain,

— $\mathbb{G}$ is a consistent finite set of literals representing the system's goals.

A *solution* to a preference-based planning problem is an applicable sequence of actions such that when executed in an initial state, it leads to a state that satisfies the conditions in $\mathbb{G}$.

Example 8 *Consider the following preference-based planning problem* $T = (\Psi_4, \Delta_1, \mathbf{C}_3, \mathbf{A}_6, \mathbb{G}_8)$*, where*

— Ψ_4 *is the state presented in Example 4,*

— $(\Delta_1, \mathbf{C}_3, \mathbf{A}_6)$ *is the planning domain presented in Example 7,*

— $\mathbb{G}_8 = \{homemade\_meal\}$

A possible solution for this planning problem is the plan:

$$\mathsf{S}_1 = [\mathsf{select\_recipe}, \mathsf{search\_storage}, \mathsf{cooking}]$$

since S_1 *is a sequence of applicable actions at* Ψ_4*, and* $\mathbb{G}_8 \subseteq \Psi^{\mathsf{S}_1}$.

So far, we have presented a planning formalism that integrates preferences into the construction of plans. In particular, the approach provides the possibility of expressing contextual preferences under which the preconditions of a

specific action should be evaluated. To encode these preferences, conditional priority expressions are used, allowing the user to specify possibly different priority criteria depending on the state of the world. In the next section, we present a partial order planning algorithm that considers the conditional-preference expressions formalized above.

3.4 Argumentation in Partial Order Planning with Contextual Preferences

The formalism described above can decide whether a plan is a solution to a preference-based planning problem, but it does not describe *how* to construct such a plan for achieving the goals of a planning system. In the following, an extension of the APOP [**?**] algorithm, called P-APOP (Argumentative Partial Order Planning with Preferences), is introduced to build plans using conditional-preference expressions. We will first show an illustrative example of how a complete plan incorporating arguments and actions is obtained in P-APOP before we go into the algorithm specifics in Section 3.5.

The P-APOP algorithm has as input the system's goals and an initial state, and outputs a partial-order plan that is a solution for the planning problem. That is, the planner starts with an initial partial plan consisting of a start step whose effects encode the initial state and a finish step whose preconditions encode the goals to be achieved, in the sense of aiming at having them warranted through the argumentation process. The initial plan is then incrementally completed with new steps until all the preconditions of these steps are warranted. Intuitively, this process generates a new partial plan whenever a new step is considered. Two types of steps are identified: *action steps*, that represent the execution of an action, and *argument steps*, that provide arguments to support the preconditions of some action step. Unlike actions, arguments are not only used to support some plan step, but they are also used to interfere or support other arguments in the plan.

Figure 3 shows a sequence of partial plans and how a complete plan is obtained by means of actions and arguments for the preference-based planning problem $(\Psi_4, \Delta_1, \mathbf{C}_3, \mathbf{A}_6, \mathbb{G}_8)$ presented in Example 8. The preconditions of the finish step represents the system's goals $\mathbb{G}_8$ and the effects of the start step encode the initial state Ψ_4. In the graphical representation, action steps are depicted by square nodes labeled with the action name. The literals appearing below an action step represent the action's preconditions, and the literals appearing above represent its effects. Moreover, the literal that appears on the right-hand side of an action step represents the selected priority criterion obtained from the conditional-preference expression associated with the action. Argument steps are represented by triangles labeled with the argument name. The literal at the top of the triangle is the conclusion of the argument. On the other hand, the solid arrows represent *causal links* of the plan, and they are used to link an effect of an action step with a precondition of another action step or with a literal in the base of an argument step. The solid arrows that link the conclusion of an argument step and a precondition of an action step

represent *support links* of the plan. The *ordering constraints* are represented by dashed arrows. These constraints allow an order to be established between steps, whereas causal and support links allow to identify the source of each literal in a plan.

In Figure 3-(a), the finish action step has one unsatisfied precondition (*homemade_meal*). The action cooking is the only one available that can be used to satisfy this precondition. Thus, cooking is added (Fig. 3-(b)) to the plan by the planning process, and its precondition becomes a subgoal to be achieved. Observe that *ing_ready* is achieved by two action steps: search_storage and receive_food_products. If search_storage (Fig. 3-(c)) is chosen, a new step is added; now, *recipe*(*pastaPuttanesca*) and *storage*(*pastaPuttanesca*) must be satisfied as preconditions. The literal *recipe*(*pastaPuttanesca*) is satisfied by the start step, but none of the available actions achieve *storage*(*pastaPuttanesca*), and from the rules in $\Delta_{\mathsf{pref\_rocio}}$, it is not possible to build an argument for *storage*(*pastaPuttanesca*) either. In that case, the algorithm fails in finding a step to achieve a subgoal, so the control is returned to the point in the algorithm where the choice was made. Now receive_food_products (Fig. 3-(d)) is chosen, a new step is added, and the precondition *food_prod_ordering* must be satisfied. The literal *food_prod_ordering* is consequence of the action order_food_products; therefore, the corresponding step order_food_products (Figure 3-(e)) is added to the plan. The literals *recipe*(*pastaPuttanesca*) and *superM*(*superfour*) are satisfied by the start step, and from the rules in $\Delta_{\mathsf{pref\_rocio}}$, where pref_rocio is the priority criterion used after evaluating E_1, it is possible to build argument $\langle \mathcal{A}_1, (suggest(superfour); 0.6)\rangle$ supporting ($suggest(superfour)$; 0.6), as well as $\langle \mathcal{A}_2, (\sim suggest(superfour); 0.4)\rangle$, which attacks the former (for the detailed structure, see Example 2). Then, argument $\langle \mathcal{A}_1, (suggest(superfour); 0.6)\rangle$ is selected since it has a greater weight and the literal in the base of the rule body conforming $\mathcal{A}_1$ is achieved by the start step, and thus a plan is finally formulated.

Note that if the criterion pref_aldo were selected, order_food_products would not be applicable since $\mathcal{A}_2$ would have a greater weight than $\mathcal{A}_1$, and under this criterion *suggest*(*superfour*) would become non-warranted – consequently, no solution plan could be formulated. This exemplifies the fact that the criterion selected for a specific action and the information considered for such selection are very relevant in the argumentative reasoning process, since the use of a different criterion can change the warrant state of an action's preconditions, clearly affecting its applicability.

As a final remark, note that the planning process does not establish a single specific sequence of actions, but rather focuses on defining a set of ordering constraints, specifying which actions must be executed before others. To determine whether a partial-order plan is a solution for a preference-based planning problem, it is necessary to first establish a correspondence between partially- and totally-ordered plans by applying a topological sorting to derive a total-order solution, as usual in the partial-order planning paradigm. Given a totally-

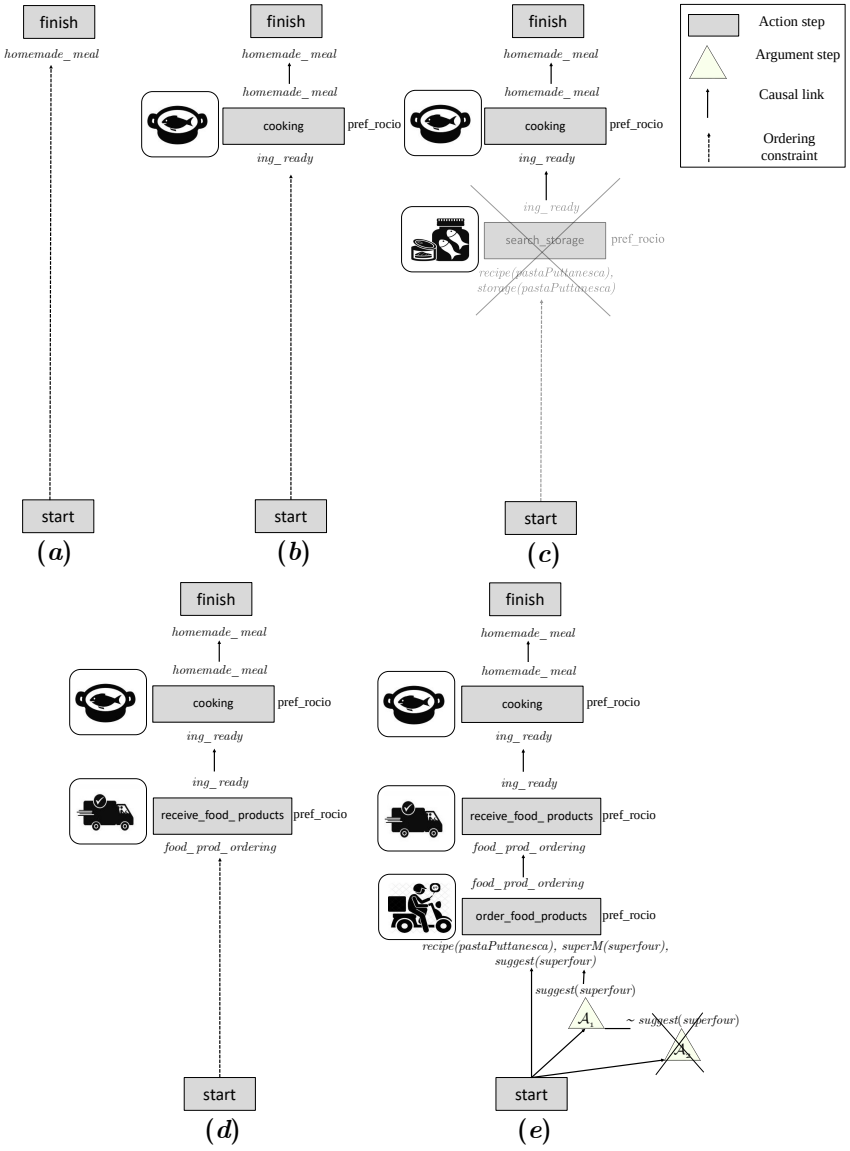


Figure 3. Partial plans for Example 8.

ordered sequence of action steps Seq derived from a particular partial plan, where each action step is consistent with the ordering constraints of the corresponding plan, we will denote with $Plan(Seq) = [\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ the sequence of actions obtained by replacing each action step in Seq with its corresponding action. Note that start and finish steps are not included in $Plan(Seq)$ because they do not correspond to the execution of any action – they are only required to represent the initial state and goals of the problem. Finally, a partial-order plan is a solution to a preference-based planning problem T when the sequence of actions $Plan(Seq) = [\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ is a solution to T.

3.5 The P-APOP Algorithm

In P-APOP, finding a partial plan consists in completing a plan by adding steps to achieve goals, as illustrated in Figure 3. For a better understanding of the P-APOP algorithm, in this section we operationally describe its main algorithms.

The P-APOP algorithm starts with an initial plan and seeks to complete it with new steps, attempting to resolve the threats that could appear. These threats appear when the effect of a new action added in the plan is to delete a literal satisfying a precondition already solved by other action steps. In this sense, when involving actions and arguments to construct plans, different types of threats can arise. In [**?**], the authors identify different types of threats that could arise in argumentation-based planning and propose methods to resolve each of them. A new type of interference is introduced in [**?**] when conditional expressions are used: an action might have interferences with the guards appearing in these expressions. Thus, the P-APOP algorithm first builds a null plan, which consists of six empty sets containing: action steps, argument steps, ordering constraints, subgoals, causal links, and support links. Then, it attempts to complete it with the recursive procedure COMPLETE_PLAN (see Algorithm 1) until all the steps' preconditions are warranted.

To achieve its goal, besides the initial state Ψ and the goals $\mathbb{G}$, function COMPLETE_PLAN considers Δ and $\mathbf{A}$ as input parameters. The set Δ contains defeasible rules whose weights will be possibly changed by the use of a different priority criterion when new action steps are added to the plan. For convenience, it is assumed that the initial weights of the rules in Δ are provided by a certain distinguished priority criterion in the set $\mathbf{C}$ of criteria the system works with. The procedure COMPLETE_PLAN begins with an unsatisfied subgoal; then, it is necessary to identify those steps that can be used to achieve such a subgoal. Towards this end, the procedure GET_STEPS is in charge of building plan steps to support an unsatisfied subgoal. The set *Steps* contains either actions in $\mathbf{A}$, or a set of argument steps for *SubGoal* built from Δ_{prc} under a given criterion prc. If no argument can be built, then only actions are considered. Note that if the algorithm fails in finding a step to achieve a subgoal, the backtracking point is updated and the control is returned at the point in the algorithm where a step choice (statement **choose**) was made.

Once the set *Steps* has been built, a step is chosen and the sets included in

Plan are updated. As we have already mentioned, after a new step is added to the plan, new threats could occur. The procedure RESOLVE_PLAN will consider all steps in the plan to detect possible interference cases that can appear and try to resolve each of them. This particular function checks four different types of threats involving arguments and actions:

- *action-action*: A precondition L is threatened by an action step A if the complemented literal $\overline{L}$ is an effect of A.
- *action-argument*: Let $\langle \mathcal{A}, (L; \alpha) \rangle$ be an argument supporting a precondition of an action step A_j; an action step A_i threatens the argument $\langle \mathcal{A}, (L; \alpha) \rangle$ if an effect of A_i negates any literal present in the set of all literals that appear as bodies of rules in the argument $\langle \mathcal{A}, (L; \alpha) \rangle$. Step A_i comes before $\langle \mathcal{A}, (L; \alpha) \rangle$.
- *argument-argument*: Let $\langle \mathcal{A}, (L; \alpha) \rangle$ be an argument added to a plan to support the precondition of an action step A; then, $\langle \mathcal{A}, (L; \alpha) \rangle$ is threatened by an argument $\langle \mathcal{B}, (Q; \beta) \rangle$ if $\langle \mathcal{B}, (Q; \beta) \rangle$ is a defeater for $\langle \mathcal{A}, (L; \alpha) \rangle$, and $\langle \mathcal{B}, (Q; \beta) \rangle$ is ordered to appear before $\langle \mathcal{A}, (L; \alpha) \rangle$ in the plan.
- *guard-action*: Let E be a conditional-preference expression. The guard γ in E is threatened by an action step A if one of its effects negates a literal present in γ, and the satisfaction state of such a guard becomes non-satisfied.

Different threat resolution methods may be applied for each kind of threat, such as including new ordering constraints for moving the cause of the threat to a harmless position or eliminating the threat with a counterargument or a new action step. A detailed description of the algorithms that deal with these problems can be found in [**?**; **?**; **?**]. Note that unresolved threats involve backtracking, which implies removing the last added step from *Plan*, and considering pending alternatives. The basic idea behind P-APOP is to search through a plan space, which can be characterized as a tree where each node represents a partial-order plan. If a failure occurs, the algorithm backtracks to the parent node. Note that the rollback process involved in the backtracking step requires identifying any links, ordering constraints, subgoals, and dependency tree associated with the failed step, and removing them without changing the rest of the plan.

Progress through the P-APOP algorithm consists of analyzing partially complete plans and modifying them in a way that brings them closer to a solution. It is easy to see that any linear plan that satisfies a partially ordered plan is such that all action step preconditions are necessarily satisfied, and backtracking ensures that the search space is eventually exhausted.

Summary of Complexity Results. After showing how the P-DeLP system can be considered together with partial-order planning techniques to consider arguments as planning steps, and sketching the P-APOP algorithm for constructing plans, we now focus on a major issue that arises in plan construction

Algorithm 1 Function to complete plan

```
1: function COMPLETE_PLAN(Plan, Δ, A, Ψ): Plan
2:     if Plan.SubGoals = ∅ then return Plan;
3:     end if
4:     choose SubGoal from Plan;
5:     Steps := GET_STEPS(SubGoal, Plan, Δ, A, Ψ);
6:     if Steps = ∅ then fail;
7:     end if
8:     choose Step from Steps;
9:     if Step is an action step OR Step is an argument step then UPDATE Plan
10:    end if
11:    RESOLVE_THREATS(Plan, Δ);
12:    COMPLETE_PLAN(Plan, Δ, A, Ψ);
13: end function
```

processes, which is related to the computational requirements that they must satisfy. Thus, understanding the inherent complexity of the reasoning tasks is crucial towards efficient implementations of defeasible argumentation-based planning systems. In that respect, the following decision problem is studied:

> *Does there exist a plan P such that, executed starting in state Ψ, arrives at goal $\mathbb{G}$ following priorities $\mathbf{C}$ and satisfies the constraints imposed by Δ and $\mathbf{A}$?*

Here we summarize from [**?**] both data and combined complexity results of query answering in the context of P-DeLP in order to analyze the difficulty of resolving preference-based planning problems under a variety of conditions. These results are based in turn on the work of [**?**], where the author provides several complexity results for SATPLAN – the decision problem of establishing whether an instance of *propositional* planning is satisfiable – and several of its restricted versions. These results, in combination with the those for DeLP reported in [**?**; **?**], are leveraged for this analysis.

Figure 4 illustrates several complexity results for a hierarchy of different planning problems, and shows how the computational complexity varies from PTIME to EXPTIME, depending on different restrictions that can be considered. The results for SATPLAN are summarized in the first complexity column reproduced from [**?**], whereas the main data and combined complexity results for P-DeLP (the decision problem of whether a literal is warranted) are given in the second column, and they are direct consequences of those in [**?**; **?**]. Finally, the third column gives the complexity results for the preference-based planning problem under each set of restricted versions. For a more detailed discussion on these results, see [**?**].

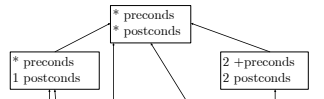
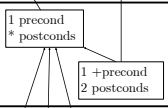
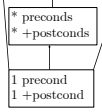
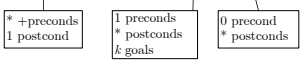
Hierarchy of Planning Problems	Complexity		
	Planning	**P-DeLP**	**Preference-based planning**
	PSPACE-complete	EXPTIME [combined]	EXPTIME, PSPACE-hard [combined]
	NP-hard	co-NP-hard [combined]	DP-hard [combined]
	NP-complete	NP [data]	NP-complete [data]
	PTIME	PTIME [data]	PTIME [data]

Figure 4. Overview of complexity results for a variety of problems (figure reproduced from [**?**]).

4 Conclusions and Perspectives

In this chapter, we have been mainly concerned with a defeasible argumentation-based approach to epistemic planning. This is a relatively recent field involving aspects of automated planning, knowledge representation, and defeasible reasoning. In order to develop theoretical formalisms and planning systems that are both expressive and practically efficient, it is necessary to combine the state of the art from all these areas. Over the years, for instance, particular attention has already been paid in the literature to efforts towards capturing more and more complex and challenging planning settings than the classical one such as planning under uncertainty, with preferences or multi-agent planning.

More specifically, in this chapter we have first provided main motivations for the need to use argumentation in the context of planning systems. Then, we have given an overview of relevant works on argumentation-based epistemic planning, focusing particularly on approaches arising in four research fields: practical reasoning, automated planning, multi-agent planning, and explainable planning. This was followed by a discussion of the use of preferences in both defeasible argumentation and planning formalisms. Finally, we presented an overview of a specific preference-based planning system, which combines partial order planning with defeasible reasoning.

Looking forward, we can identify several topics and research directions in the area of defeasible planning. We now briefly discuss some of the ones we find particularly interesting:

- The study of planning processes as search through a space of plans naturally lends to analytical studies of computational complexity. Analyses of aspects related to efficiency in the use of resources are of great importance, but little progress has been made in this direction in argumentation-based planning. Much work can still be done in the realm of algorithms and complexity, such as studying sub-problems for which efficiency guarantees can be established.

- Incorporating *humans in the planning loop*, especially in collaborative settings, presents several important challenges that must be addressed. Explainable planning seeks to build trust and transparency when interacting with humans; thus, even though explainability is not a new topic, it is another promising research line in the realm of epistemic planning.

- While defeasible planners have been effectively applied in different domains, it is important to achieve implementations with empirical evaluations in real settings, and compare obtained results with other approaches from the literature in terms of effectiveness and efficiency. There are many opportunities in this line of work.

- The P-APOP algorithm does not leverage heuristics, so another promising direction to analyze in the future is the adaptation of the algorithm to include different heuristic methods that allow the reduction of the search space and, consequently, the overall computational cost.

- In *abstract* argumentation frameworks, the specification of different *semantics* encodes different criteria of acceptability of (sets of) arguments. Establishing a correspondence between our framework and such argumentation semantics, and their associated properties, is a promising line of work that can serve to investigate how the selection of the best plan(s) can be based on well-established properties.

- Conditional-preference expressions constitute a key component in the argumentation-based planning framework we have described in Section 3 to decide which actions to keep while constructing plans. A detailed study of such expressions and properties that characterize them for rational decision-making is an interesting open task for future research.

- Another important topic is the relationship between the notion of threat and attack present in the argumentation literature. On the other hand, it would also be especially interesting to study how to incorporate values into defeasible rules based on rationality principles, like the ones proposed by [**?**]. This is a challenging objective for future research.

- Finally, the study of other preference representation models and tools – such as operators for combining or prioritizing contexts – is also a challenge that deserves attention.

These research directions are only a few of the ones that stand out; the goal of this chapter was to describe an area that shows promising early results, but with many opportunities for both basic and applied research and development.

Acknowledgements

The authors are indebted to Guillermo R. Simari for inspiring discussions.

This work was funded in part by Universidad Nacional del Sur (UNS) under grants PGI 24/ZN057 and PGI 24/N055, Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) under grant PIP 11220210100577CO, Agencia Nacional de Promoción Científica y Tecnológica, Argentina under grant PICT-2018-0475 (PRH-2014-0007), and Universidad Nacional de Entre Ríos (UNER) under grant PDTS-UNER 7066). Godo acknowledges partial support by the Spanish project LINEXSYS (PID2022-139835NB-C21) funded by MCIN/AEI/10.13039/501100011033.

Juan C. L. Teze
Fac. de Cs. de la Adm., Universidad Nacional de Entre Ríos (UNER)
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Email: carlos.teze@uner.edu.ar

Lluis Godo
Artificial Intelligence Research Institute (IIIA)
Spanish National Council for Scientific Research (CSIC)
Email: godo@iiia.csic.es

Gerardo I. Simari
Department of Computer Science and Engineering, Universidad Nacional del Sur (UNS), Institute for Computer Science and Engineering (ICIC UNS-CONICET), Argentina, and School of Computing and Augmented Intelligence, Arizona State University, USA
Email: gis@cs.uns.edu.ar