An Approach to Improve Argumentation-Based Epistemic Planning with Contextual Preferences

Juan C. L. Teze[a,d,*], Lluis Godo[b], Gerardo I. Simari[c]

[a] *Grupo de Investigación en Agentes y Sistemas Inteligentes (GINAySI), Fac. de Cs. de la Adm., Universidad Nacional de Entre Ríos (UNER) Monseñor Tavella 1424, (3200) Concordia, Entre Ríos, Argentina*
[b] *Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB – 08193 Bellaterra, Barcelona, Spain*
[c] *Dept. of Computer Science and Engineering, Universidad Nacional del Sur (UNS) and Institute for Computer Science and Engineering (ICIC UNS-CONICET), San Andrés 800, (8000) Bahía Blanca, Buenos Aires, Argentina*
[d] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) Av. Rivadavia 1917, (C1033AAJ) Ciudad Autónoma de Buenos Aires, Argentina*

Abstract

Current approaches to argumentation-based planning represent an interesting proposal where defeasible argumentation is used as a practical mechanism suitable for reasoning with potentially contradictory information in dynamic environments. In many real-world planning scenarios, the development of formalisms allowing explicit preference specification over pieces of knowledge turns out to be an essential task—however, despite its importance, existing planning systems are not provided with the possibility of dynamically changing these preferences when a plan is being constructed. This paper presents an argumentation-based approach to deal with the handling of preferences when a plan is formulated; in particular, we propose using conditional expressions to select and change priorities regarding information upon which plans are constructed. Our aim is not to improve the efficiency of current planning systems, but to enhance the resulting plan itself by introducing an approach capable of representing and handling multiple preferences over defeasible knowledge. This approach will contribute to the strengthening of existing argumentation-based epistemic planning systems, providing a useful tool that the user could exploit. Finally, we also present a running-time analysis and several complexity results associated with our approach.

Keywords: Planning, Defeasible argumentation, Preferences

*Corresponding author at: Fac. de Cs. de la Adm., Universidad Nacional de Entre Ríos Monseñor Tavella 1424, (3200) Concordia, Entre Ríos, Argentina. Tel.: +543454231400.
Email address: carlos.teze@uner.edu.ar (Juan C. L. Teze)

1. Introduction

Planning is a research area of AI that addresses the problem of obtaining a set of actions to achieve a specific goal given an initial state of the world. Recently, the consideration of *epistemic* elements in building a plan has extended the area: "*Epistemic planning is the enrichment of planning with epistemic notions, that is, knowledge and beliefs*" [1, 2, 3]. Defeasible argumentation is a form of reasoning about beliefs that can be used to exploit knowledge bases in the context of possible inconsistencies [4, 5]. 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 and then comparing those in conflict to reach a decision regarding acceptance. Defeasible argumentation-based epistemic planners have been adequately applied in the context of complex and dynamic domains where contradictory and incomplete information is considered [6, 7, 8]; these planners are characterized by the efficient use of defeasible reasoning for the epistemic tasks performed over the represented knowledge. In particular, defeasible argumentation is used as the inference mechanism to reason about the preconditions and effects of a planner system's actions.

In many planning approaches [9, 10, 11], 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. However, and despite its importance in the reasoning process, existing argumentation-based planning systems do not provide additional capabilities for dynamically changing the preference expressed by these priorities when a plan is being constructed. The main contribution of this paper centers on revising and extending previous work [12] by introducing an epistemic planning formalism to equip a planner's reasoning engine with tools for carrying out this task. In particular, we will deal with the problem of preference handling by introducing a way to select a priority assignment mechanism to modify the preferences among different pieces of defeasible knowledge in each step (action) of a plan; in order to formalize such selection, we will use the conditional expression structure proposed in [13].

In formulating our proposal, we will start from the proposal in [6, 7], where a DeLP-based epistemic planner was first introduced. Concretely, this formalism presents an approach where actions are selected and incorporated into a plan after argumentation-based epistemic reasoning is performed. In this paper, we will use the P-DeLP framework belonging to the DeLP family, which will provide the inference engine for reasoning over the knowledge used by the planner—its language and the reasoning mechanism will be described in Section 3. P-DeLP stands for *Possibilistic Defeasible Logic Programming*, and considers weights associated with the object language. In contrast to our approach, [6, 7] focus on constructing plans using partial order planning techniques, whereas we focus on developing a framework to deal with the problem of constructing plans based on multiple preferences at the defeasible knowledge level through weights. In our approach, arguments are sets of weighted formulas that support a conclusion, and such weights are used to compute an argument's strength and then settle conflicts among contradictory conclusions. In analyzing the applicability of a particular action, our proposal provides a way to specify conditional expressions, which will to help decide what preference relation should be considered in each context. We will introduce an extension of the APOP algorithm presented in [7] that considers these expressions. Part of our work is also focused on studying aspects related to computational complexity in the context of our approach.

The rest of the paper is structured as follows. Section 2 introduces a running example to motivate our proposal's main ideas. In Section 3, we give an overview of Possibilistic Defeasible Logic Programming (P-DeLP), while in Section 4 we formalize our mechanisms for expressing preferences. Then, in Section 5, we present our characterization of a defeasible argumentation-based epistemic planner with the distinctive capability of handling preferences. We extend the APOP algorithm presented in [7] with our approach, and in Section 6 we analyze a new type of interference that can arise in a plan. Section 7 presents a running time analysis of the main processes used by our algorithm and gives several complexity results for our planning problem. In Section 8, we introduce an example exercising our proposal. Finally, in Section 9 we include the relevant related work, and in Section 10 we offer our conclusions and possible future research lines.

2. Motivation

An argumentation-based epistemic planner traditionally adopts a single perspective in the construction of a plan, which is reflected by the strategy used in the process of establishing why an argument is favored over others. These strategies are established to compare information and decide between conflicting arguments capturing the importance or priority of the information that arguments contain. Given the usual dynamic nature of preferences, designing a real planning system with a single perspective when facing different situations is quite limiting. Furthermore, when the comparison fails—it is indecisive—the system could become ineffective because it fails in its primary role of making a decision. This situation could improve with the expansion of this type of planner to consider context-adaptable priorities [14]; that is, priorities given in terms of defeasible rules attached with weights that may vary from one particular context to another. A possible way to address such a circumstance is to consider tools that allow to dynamically change the information priority criterion to be applied as the planner reasons and chooses which actions to add to a plan.

The use of a priority criterion can be mainly understood as a means for modeling the choices of users. Instead of considering a single criterion to prioritize defeasible knowledge, we propose that the system evaluate the situation considering several contexts and decide which one to use depending on the current state of the world. Consider, for instance, a scenario in which a planner system considers the following actions: recommend tourist destinations, suggest airlines, and advise adopting travel insurance policies. To find a plan for giving a recommendation, the system could consider the user's particular preferences through explicitly-specified priorities, which are obtained in advance. Suppose the system can use two priority criteria; one based on *trust* and another based on *price*. For illustration purposes, let us also assume that the system considers the following domain knowledge about travel insurance policies:

Rule	**Description**
r_1	An insurance policy is not suggested if it is expensive.
r_2	An expensive policy is suggested if it is trusted by a user.

Considering the *price*-based criterion, r_1 will have higher priority since the price of what is being offered is under consideration, whereas r_2 is prioritized if the user's *trust*-based criterion is considered. Then, the user can inform in which situations (context) to use each one by specifying conditions; for instance, when the offered location is one of the most expensive places for tourists, then perhaps the *price* should be used; otherwise, the *trust*-based criterion is used. Considering a conditional criterion selection allows these systems to adopt the criterion that better adapts to their knowledge of the situation. As we can see, this new approach provides a different way to solve planning problems by using more than one preference relation in the planning process by considering the system's epistemic states as they are affected by their environment. Hence, our approach focuses on two issues: how to use multiple priority criteria and how to decide the applicability of an action leveraging conditional selection statements over prioritized knowledge.

3. Preliminaries

In solving a planning problem, planning systems should be provided with an appropriate set of actions whose representation must consider all the preconditions and effects that are relevant to solve the problem. In many real-world applications, these systems often have to deal with potentially contradictory and incomplete knowledge about the environment. In this context, structured argumentation has played a significant role in capturing and representing this type of knowledge to be used in reasoning about actions. As we will explain later, the preconditions to execute actions can be satisfied by other actions' effects (as usual) or by conclusions supported by arguments that are based on inference rules and other actions' effects. We will also show that apart from effects that are present in the definition of actions, there could be more effects that the system will be able to deduce using argumentation-based reasoning.

In this section, we present a structured argumentation framework based on the mechanisms of *Defeasible Logic Programming* (DeLP)–see [15] for full details on DeLP. The main difference between the framework we will introduce and DeLP lies in the use of *Possibilistic Logic* (PL) [16] as labels adorning the formulas with a necessity degree at the object level of the language. The inclusion of the labels permits the consideration of preferences in this formalization; we refer to this structured argumentation system as *Possibilistic Defeasible Logic Programming* (P-DeLP). We will summarize the elements we require here from the works in [17, 16, 18][1]; the interested reader is referred to these works for a thorough presentation. A variant of P-DeLP with a somewhat different semantics can be found in [19].

3.1. Background on Possibilistic logic

Possibilistic Logic (see *e.g.,* [20] 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 [21, 22, 23], and this is the reason we have chosen it for our argumentation-based planning framework. We will now give a bare-bones introduction of this uncertainty logic, as necessary for our presentation.

Possibilistic logic was initially defined on top of classical logic, although there are generalizations that use a substratum of non-classical or fuzzy logics. Here we will stick to the classical setting. We start from a classical propositional language $\mathcal{L}$, built from a (countable) set of propositional variables $\{p_1, p_2, \ldots\}$ and Boolean connectives $\wedge$, $\vee$, and $\neg$. Let us denote by $\mathfrak{I}$ the set of all Boolean interpretations $\mathbb{I} : \mathcal{L} \to \{0, 1\}$ over the language $\mathcal{L}$, defined in the usual way. For each proposition ϕ we will denote by $[\phi]$ the set of its models, i.e. $[\phi] = \{\mathbb{I} \in \mathfrak{I} \mid \mathbb{I}(\phi) = 1\}$.

Possibilistic logic attaches weights from the unit interval $[0, 1]$ to propositions to model statements of the form "ϕ is certain at least to the degree r", where ϕ is a proposition of $\mathcal{L}$ that represents an (imprecise) knowledge about the state of the world, and $r \in [0, 1]$ represents a lower bound on the belief of ϕ in terms of necessity measures, with the following convention: if $r = 1$ it means that ϕ is known to be fully certain, the higher is r the more certain is ϕ, and if $r = 0$ it means nothing is known about ϕ. Necessity values can also be interpreted as degrees of preference or levels of priority when dealing with goals [21, 24]. Weighted formulas will be denoted as pairs (ϕ, r).

Belief states in the possibilistic view of uncertainty are modeled as *possibility distributions* on the set $\mathfrak{I}$ of all possible interpretations required to be *normalized*, *i.e.,* at least one interpretation must be *totally* possible. A (normalized) possibility distribution is thus a mapping $\pi : \mathfrak{I} \to [0, 1]$ such that $\pi(\mathbb{I}) = 1$ for some $\mathbb{I} \in \mathfrak{I}$, assigning a degree of plausibility to every possible interpretation. Here $\pi(\mathbb{I}) = 1$ means that $\mathbb{I}$ is a fully plausible interpretation, $\pi(\mathbb{I}) = 0$ means that $\mathbb{I}$ is a completely discarded interpretation, and in general, the higher is $\pi(\mathbb{I})$, the more plausible is $\mathbb{I}$. Then, to measure the certainty level induced on

[1] Actually, the framework developed in these papers is more general than the one considered here, in the sense that the underlying logic in these references was assumed to be Gödel fuzzy logic, while here we will deal with classical propositional logic.

a proposition ϕ by such a possibilistic model π, it is customary to compute the necessity and possibility measures of the set $[\phi]$ of interpretations satisfying ϕ, according to the following definitions [25]:

$$N_\pi(\phi) = \inf_{\mathbb{I} \in \mathfrak{I}} \{1 - \pi(\mathbb{I}) \mid \mathbb{I} \notin [\phi]\}, \quad \Pi_\pi(\phi) = \sup_{\mathbb{I} \in \mathfrak{I}} \{\pi(\mathbb{I}) \mid \mathbb{I} \in [\phi]\}$$

$N_\pi(\phi)$ estimates to what extent all the counter-models of ϕ (i.e., the models of $\neg\phi$) are implausible, while $\Pi_\pi(\phi)$ estimates to what extent there is at least a model of ϕ that is plausible. Necessity and possibility measures are dual in the sense that $N_\pi(\phi) = 1 - \Pi_\pi(\neg\phi)$, hence expressing the fact that a proposition ϕ is certain to the extent in which $\neg\phi$ is deemed implausible. Necessity and possibility measures satisfy these characteristic decomposability properties:

$$\begin{aligned} N_\pi(\phi \wedge \psi) &= \min(N_\pi(\phi), N_\pi(\psi)) \\ \Pi_\pi(\phi \vee \psi) &= \max(\Pi_\pi(\phi), \Pi_\pi(\psi)) \end{aligned}$$

for any pair of propositions ϕ, ψ, and $N_\pi(\bot) = 0$ (or equivalently $\Pi_\pi(\top) = 1$). As a consequence, the following condition is also satisfied for any ϕ: if $N_\pi(\phi) > 0$ then $N_\pi(\neg\phi) = 0$. Therefore, if $N_\pi(\phi) = 1$ then ϕ is considered a fully certain piece of information, if $N_\pi(\neg\phi) = 1$ then ϕ is fully disbelieved, while the case $N_\pi(\phi) = N_\pi(\neg\phi) = 0$ represents full ignorance about ϕ.

In Possibilistic logic, a weighted formula (ϕ, r) will be satisfied by a possibilistic model π whenever $N_\pi(\phi) \geq r$. Then, the following weighted version of the resolution rule is sound in possibilistic logic:

$$\frac{(\phi \vee \psi, r) \quad (\neg\phi \vee \chi, t)}{(\psi \vee \chi, \min(r, t))}$$

Actually, for propositions in clausal form, this rule was shown to be also complete for deduction by refutation—see [20] for details.

Possibilistic logic has also been adapted to qualify conditional formulas "if ψ then ϕ" rather than material implications $\neg\psi \vee \phi$ by using conditional measures. In contrast to probability theory, where there is a predominant way of expressing the conditional probability $Pr(\phi \mid \psi)$ of ϕ given ψ simply by means of the ratio $Pr(\phi \wedge \psi)/Pr(\psi)$, in the case of possibility theory there have been different proposals. Among them, the most purely qualitative one regarding possibility measures is taking the conditional possibility $\Pi_\pi(\phi \mid \psi)$ as the maximum solution x of the equation $\min(x, \Pi_\pi(\psi)) = \Pi_\pi(\phi \wedge \psi)$—see [26]. By duality, taking $N_\pi(\phi \mid \psi) = 1 - \Pi(\neg\phi \mid \psi)$ leads to the following definition of conditional necessity:

$$N_\pi(\phi \mid \psi) = \begin{cases} 0, & \text{if } N_\pi(\neg\psi) = N_\pi(\neg\psi \vee \phi) \\ N_\pi(\neg\psi \vee \phi), & \text{otherwise} \end{cases}$$

which is a slight variant of $N_\pi(\neg\psi \vee \phi)$. With this definition, one can check that the following weighted version of the resolution rule still holds:

$$\frac{N_\pi(\phi \mid \psi) \geq r \quad N_\pi(\psi \mid \chi) \geq t}{N_\pi(\phi \mid \chi) \geq \min(r, t)},$$

which, in turn, also validates the following (conditional) possibilistic modus ponens inference rule:

$$\frac{N_\pi(\phi \mid \psi_1 \wedge \ldots \wedge \psi_n) \geq r, \quad N_\pi(\psi_1) \geq t_1, \ldots, N_\pi(\psi_n) \geq t_n}{N_\pi(\phi) \geq \min(r, t_1, \ldots, t_n)} \qquad [PMP]$$

This rule will be used in the P-DeLP framework described in the next section.

3.2. P-DeLP: Language and Reasoning Mechanisms

In P-DeLP, formulas will be supported by arguments, which will have an attached weight (formally a necessity degree) associated with the supported conclusion. The ultimate answer to queries will be 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 [15] and will be presented below.

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 $N(L \mid L_1 \wedge \ldots \wedge L_k)$ in the case $R = L \leftarrow L_1, \ldots, L_k$, or a lower bound for the necessity degree $N(L)$ in the case $R = L$. Note that, by considering $N(L \mid L_1 \wedge \ldots \wedge L_k)$ we are following the usual notational conventions in Logic Programming [27] that regard the set of literals in the body of a clause $L_1, \ldots, L_k$ as a conjunction of these literals. Also, following [15], P-DeLP rules can also be represented as schematic rules with variables; as usual in Logic Programming, schematic variables are denoted with an initial uppercase letter. To keep the usual terminology in defeasible reasoning, we distinguish between *strict* and *defeasible* clauses: a clause $(R; \omega)$ is referred as strict if $\omega = 1$ (top priority) and defeasible otherwise, i.e. if $\omega < 1$. Of course, the higher is weight ω, the higher is 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. Continuing with the running example of Section 2, let $\mathbb{P}_1 = (\Pi_1, \Delta_1)$ be a P-DeLP program that represents information for recommending travel insurance policies.

$$\Pi_1 = \left\{ \begin{array}{l} (\mathit{airline}(a1, \mathit{ana});\ 1) \\ (\mathit{spendIns}(1500, \mathit{ana});\ 1) \\ (\mathit{costIns}(1700, i1);\ 1) \\ (\mathit{lostLuggage}(i1);\ 1) \\ (\mathit{insurance}(I) \leftarrow \mathit{expectTrouble}(I);\ 1) \end{array} \right\}$$

$$\Delta_1 = \left\{ \begin{array}{l} (\sim insurance(I,U) \leftarrow expIns(I,U);\ 0.2) \\ (insurance(I,U) \leftarrow expIns(I,U), trustIns(I,U);\ 0.1) \\ (trustIns(I,U) \leftarrow airline(A,U), lostLuggage(I);\ 0.85) \\ (trustIns(I,U) \leftarrow travelAg(T,U), insT(I,T);\ 0.8) \\ (expIns(I,U) \leftarrow spendIns(E,U), costIns(C,I), C > E;\ 0.95) \\ (\sim expIns(I,U) \leftarrow spendIns(E,U), costIns(C,I), C < E;\ 0.1) \\ (\sim prefA(A,U) \leftarrow expA(A);\ 0.6) \\ (prefA(A,U) \leftarrow expA(A), trustA(A,U);\ 0.3) \\ (\sim prefTAg(T,U) \leftarrow expTAg(T);\ 0.7) \\ (prefTAg(T,U) \leftarrow expTAg(T), trusTAg(T,U);\ 0.2) \\ (trustA(A,U) \leftarrow trust(U,U1), trustA(A,U1);\ 0.2) \\ (expA(A) \leftarrow topRatingA(A);\ 0.3) \\ (trusTAg(T,U) \leftarrow trust(U,U1), trusTAg(T,U1);\ 0.9) \\ (expTAg(T) \leftarrow topRatingTA(T);\ 0.6) \end{array} \right\}$$

Observe that the set Π_1 of clauses considered strict has four facts and one rule. These facts represent information that can be automatically obtained by a planner system: $a1$ is an airline recommended for ana ($airline(a1, ana)$), ana hopes to spend no more than 1500 on travel insurance ($spendIns(1500, ana)$), 1700 is the cost of the travel insurance policy $i1$ ($costIns(1700, i1)$), and $i1$ covers the cost of lost luggage ($lostLuggage(i1)$). The remaining strict rule in Π_1 contains the advice that if trouble is expected with the airline, then travel insurance should be obtaind. As ana is not expecting trouble, this fact is not part of Π_1.

The set Δ_1 has several defeasible rules, and each one has an associated priority degree. The first two rules represent reasons for and against suggesting a travel insurance policy: if I is an expensive policy ($expIns(I, U)$), then there is a reason against suggesting it, whereas if I is trusted by a user ($trustIns(I, U)$), then there exist reasons for suggesting it. The third and fourth rules represent reasons for establishing whether a particular user trusts a policy; if I covers the cost of lost luggage or works with a recommended travel agency ($travelAg(T,U)$), then there are tentative reasons to believe that it is a trusted one. Defeasible rules 7–10 introduce some reasons for and against suggesting an airline or travel agency, and a suggestion is given if a user trusts it. The eleventh rule expresses that "a travel agency ($trusTAg(T,U1)$) which is trusted by a trusted user ($trust(U,U1)$) typically will be trusted". Finally, a travel agency T is not suggested if it is expensive. Thus, the fourteenth defeasible rule can be read as follows: "if T is one of the top agencies rated by the users ($topRatingTA(T)$), then it is expensive". □

We will use the possibilistic inference meta-relation $\mid\!\sim$ between $\mathbb{P}$ and $(L;\omega)$, *i.e.,* $\mathbb{P} \mid\!\sim (L;\omega)$, which 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*

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

where $(H \leftarrow H_1, \ldots, H_k; \beta) \in \mathbb{P}$ and all the weighted literals $(H_1; \gamma_1), ..., (H_k; \gamma_k)$ appear before $(L_i; \omega_i)$ in the sequence.

Note that this inference rule faithfully corresponds to the possibilistic modus ponens rule [PMP] introduced in the previous section, once we interpret the weights in clauses as we prescribed above.

A P-DeLP program $\mathbb{P}$ will be deemed as *contradictory*, denoted $\mathbb{P} \mid\!\sim \perp$, if, for some atom a, $\mathbb{P} \mid\!\sim (a; \omega)$ and $\mathbb{P} \mid\!\sim (\sim a; \beta)$, with $\omega > 0$ and $\beta > 0$. Given a P-DeLP program $\mathbb{P} = (\Pi, \Delta)$, we will assume that the set Π of all the clauses that are considered as strict (*i.e.,* with weight $\omega = 1$) is non-contradictory, *i.e.,* $\Pi \not\mid\!\sim \perp$.

Next, we introduce the notion of argument in the setting of a P-DeLP program.

Definition 1 (Argument). *Let $\mathbb{P} = (\Pi, \Delta)$ be a P-DeLP program. We say that a set of defeasible rules $\mathcal{A} \subseteq \Delta$ is an* argument *for a literal L with necessity degree $\omega > 0$, denoted $\langle \mathcal{A}, (L; \omega) \rangle$, if*

- (i) $\Pi \cup \mathcal{A} \mid\!\sim (L;\ \omega)$*, and* $\omega = \max\{\beta \in [0, 1] \mid \Pi \cup \mathcal{A} \mid\!\sim (L;\ \beta)\}$*, i.e., ω is the greatest degree of deduction of L from $\Pi \cup \mathcal{A}$.*
- (ii) $\Pi \cup \mathcal{A}$ *is non-contradictory,*
- (iii) $\mathcal{A}$ *is $\subseteq$-minimal, i.e., there is no $\mathcal{A}' \subsetneq \mathcal{A}$ satisfying (i) and (ii).*

An argument $\langle \mathcal{S}, (H; \gamma) \rangle$ is a sub-argument *of $\langle \mathcal{A}, (L; \omega) \rangle$ iff $\mathcal{S} \subseteq \mathcal{A}$. When no confusion may arise, we will overload the notation by writing $\langle \mathcal{S}, (H; \gamma) \rangle \subseteq \langle \mathcal{A}, (L; \omega) \rangle$, and will also simplify the notation even further by writing $\mathcal{S} \subseteq \mathcal{A}$ when the rest of the elements are understood from the context.*

Example 2. Consider the next two arguments: $\langle \mathcal{A}_1, (insurance(i1, ana); 0.1) \rangle$ and $\langle \mathcal{A}_2, (\sim insurance(i1, ana); 0.2) \rangle$, constructed from the DeLP-program $\mathcal{P}_1$ presented in Example 1, where

$$\mathcal{A}_1 = \left\{ \begin{array}{l} (insurance(i1, ana) \leftarrow expIns(i1, ana), trustIns(i1, ana);\ 0.1) \\ (expIns(i1, ana) \leftarrow spendIns(1500, ana), costIns(1700, i1), 1700 > 1500;\ 0.95) \\ (trustIns(i1, ana) \leftarrow airline(a1, ana), lostLuggage(i1);\ 0.85) \end{array} \right\}$$

$$\mathcal{A}_2 = \left\{ \begin{array}{l} (\sim insurance(i1, ana) \leftarrow expIns(i1, ana);\ 0.2) \\ (expIns(i1, ana) \leftarrow spendIns(1500, ana), costIns(1700, i1), 1700 > 1500;\ 0.95) \end{array} \right\}$$

and each literal $(insurance(i1, ana);\ 0.1)$ and $(\sim insurance(i1, ana);\ 0.2)$ is obtained by applying the inference rule *generalized modus ponens* presented above.

Given a program, it can be the case that there exist arguments supporting contradictory literals. Two literals are contradictory if they are complementary with respect to strong negation. In what follows, for a given literal L, we will write its complement as $\overline{L}$ to denote "$\sim a$" if $L = a$, and "a" if $L = \sim a$.

Given a program $\mathbb{P} = (\Pi, \Delta)$ and two weighted literals $(L_1;\, \omega_1)$ and $(L_2;\, \omega_2)$, with $\omega_1, \omega_2 > 0$, such that $\Pi \cup \{(L_1;\, \omega_1)\} \not\vdash \bot$ and $\Pi \cup \{(L_2;\, \omega_2)\} \not\vdash \bot$, we say that they *disagree* regarding $\mathbb{P}$ when $\Pi \cup \{(L_1;\, \omega_1), (L_2;\, \omega_2)\} \vdash \bot$. Two complementary weighted literals trivially disagree.

Definition 2 (Attack). *Let $\mathbb{P} = (\Pi, \Delta)$ be P-DeLP program, and let $\langle \mathcal{A}, (L;\alpha)\rangle$ and $\langle \mathcal{B}, (Q;\beta)\rangle$ be two arguments built from $\mathbb{P}$. We say that $\langle \mathcal{B}, (Q;\beta)\rangle$* attacks *$\langle \mathcal{A}, (L;\alpha)\rangle$* at the weighted literal *$(H;\, \gamma)$ if and only if there exists a subargument $\langle \mathcal{S}, (H;\gamma)\rangle$ of $\langle \mathcal{A}, (L;\alpha)\rangle$, called the* disagreement subargument*, such that $(H;\, \gamma)$ and $(Q;\, \beta)$ disagree with respect to $\mathbb{P} = (\Pi, \Delta)$.*

We also say that an argument $\langle \mathcal{A}, (L;\alpha)\rangle$ is a counterargument for another argument $\langle \mathcal{B}, (Q;\beta)\rangle$ when there is an attack from the former to the latter.

In P-DeLP, given an argument that attacks another, such as the two arguments in the previous example:

$$\langle \mathcal{A}_1, (insurance(i1, ana); 0.1)\rangle \text{ and } \langle \mathcal{A}_2, ({\sim}insurance(i1, ana); 0.2)\rangle),$$

the arguments are compared using a simple *strategy* to decide which one prevails: an argument $\langle \mathcal{A}, (L;\alpha)\rangle$ is preferred to an argument $\langle \mathcal{B}, (Q;\beta)\rangle$, denoted $\langle \mathcal{A}, (L;\alpha)\rangle > \langle \mathcal{B}, (Q;\beta)\rangle$, when $\alpha > \beta$.

The notion that an argument *defeats* another one results from a successful attack.

Definition 3 (Defeat). *Let $\langle \mathcal{A}, (L;\alpha)\rangle$ and $\langle \mathcal{B}, (Q;\beta)\rangle$ be two arguments. We say that $\langle \mathcal{B}, (Q;\beta)\rangle$* defeats *$\langle \mathcal{A}, (L;\alpha)\rangle$, or equivalently that $\langle \mathcal{B}, (Q;\beta)\rangle$ is a* defeater *of $\langle \mathcal{A}, (L;\alpha)\rangle$, if $\langle \mathcal{B}, (Q;\beta)\rangle$ attacks $\langle \mathcal{A}, (L;\alpha)\rangle$ with disagreement subargument $\langle \mathcal{S}, (\overline{H};\gamma)\rangle$, and $\beta \geq \gamma$.*

Moreover, we will say that:

– *$\langle \mathcal{B}, (Q;\beta)\rangle$ is a* proper defeater *of $\langle \mathcal{A}, (L;\alpha)\rangle$ if $\beta > \gamma$, and*

– *$\langle \mathcal{B}, (Q;\beta)\rangle$ is a* blocking defeater *of $\langle \mathcal{A}, (L;\alpha)\rangle$ if $\beta = \gamma$.*

It is interesting to note that in the case of blocking defeaters, both arguments $\langle \mathcal{A}, (L;\alpha)\rangle$ and $\langle \mathcal{B}, (Q;\beta)\rangle$ defeat each other, *i.e.,* the result of the attack is that both arguments end up being defeated.

Once we have defined the attack relation between arguments and the notion of defeat, the next step is to stipulate how a conclusion can be obtained from a knowledge base specified as a P-DeLP program. Following the DeLP approach, our procedure will dialectically analyze the arguments that can be built for a possible conclusion represented by a given weighted literal, as explained next.

Given a P-DeLP program $\mathbb{P}$ and an argument $\langle \mathcal{A}, (L;\alpha)\rangle$ built from $\mathbb{P}$, to establish whether $\langle \mathcal{A}, (L;\alpha)\rangle$ is acceptable or undefeated, all the possible defeaters for $\langle \mathcal{A}, (L;\alpha)\rangle$ are considered successively, and for each one of them, the defeaters for this defeater will be considered recursively in a dialectical process.

As each defeater could in turn be defeated, a finite sequence of arguments such as

$$\Lambda = [\langle \mathcal{A}_0, (L_0; \omega_0)\rangle, \langle \mathcal{A}_1, (L_1; \omega_1)\rangle \ldots, \langle \mathcal{A}_n, (L_n; \omega_n)\rangle],$$

where each argument (except the first one) is a defeater of its predecessor, is called an *argumentation line*. In Λ, considering the position of the arguments in the line, the ones in even positions with regard to the initial argument play a role of support, and those in odd positions act as interfering arguments. In P-DeLP, an argumentation line Λ is *acceptable* if the following conditions hold:

(1) Λ is finite,
(2) the set of supporting arguments in Λ is non-contradictory, and the set of interfering arguments in Λ is also non-contradictory,
(3) no argument $\langle \mathcal{A}_j, (L_j; \omega_j)\rangle$ in Λ is a subargument of an argument $\langle \mathcal{A}_i, (L_i; \omega_i)\rangle$ in Λ, with $i < j$, and
(4) every blocking defeater $\langle \mathcal{A}_i, (L_i; \omega_i)\rangle$ in Λ is defeated by a proper defeater $\langle \mathcal{A}_{i+1}, (L_{i+1}; \omega_{i+1})\rangle$ in Λ.

These four constraints are necessary to avoid fallacious situations, such as the generation of infinite lines or the switch of roles for a given argument, among others (see [15] for a complete discussion). We will also say that $\langle \mathcal{A}_{n+1}, (L_{n+1}; \omega_{n+1})\rangle$ is acceptable w.r.t. $\Lambda = [\langle \mathcal{A}_0, (L_0; \omega_0)\rangle, \langle \mathcal{A}_1, (L_1; \omega_1)\rangle \ldots, \langle \mathcal{A}_n, (L_n; \omega_n)\rangle]$, if $\Lambda = [\langle \mathcal{A}_0, (L_0; \omega_0)\rangle, \langle \mathcal{A}_1, (L_1; \omega_1)\rangle \ldots, \langle \mathcal{A}_n, (L_n; \omega_n)\rangle, \langle \mathcal{A}_{n+1}, (L_{n+1}; \omega_{n+1})\rangle]$ is an acceptable argumentation line.

Of course, there can be more than one defeater for a particular argument; therefore, many acceptable argumentation lines could arise from this argument, leading to a tree structure called a *dialectical tree*. Each path from the root of the dialectical tree to a leaf corresponds to a different acceptable argumentation line. Observe that every node (except the root) is a defeater of its parent and, naturally, all leaves are undefeated.

Given an argument $\langle \mathcal{A}, (L; \alpha)\rangle$, to decide whether $(L;\ \alpha)$ is *warranted*, every node in the dialectical tree associated with $\langle \mathcal{A}, (L; \alpha)\rangle$ is marked as undefeated ("U"), or defeated ("D"). All leaf nodes are marked as "U", and an inner node is marked as "D" if it has at least one child marked as "U", but it will be marked as "U" if all its children are marked as "D". Thus, a ground literal $(L;\ \alpha)$ is *warranted* from a program $\mathbb{P} = (\Pi, \Delta)$ if there exists an argument $\langle \mathcal{A}, (L; \alpha)\rangle$ for $(L;\ \alpha)$ from $\mathbb{P}$ such that the root of its associated dialectical tree is marked as "U".

In the next section, we will propose a tool that allows selecting and changing weights attached to prioritized rules. In Section 5, we will formally introduce a preference-based argumentation approach for solving planning problems. To do this, and as explained in Section 1, we will extend and refine several concepts introduced in [12] to achieve the mentioned goals.

4. Preference Handling

In the previous section, we presented the P-DeLP system that will be used in this paper to reason defeasibly during the planning process. One of our goals is to adjust the priority weights on rules to be used by P-DeLP's inference mechanism when selecting actions in the planning process. For that, we will introduce a way of specifying a context-adaptable priority assignment mechanism that relies on the use of conditional expressions, which in turn constitutes one of contributions in our approach. Before formally defining this type of expressions, we will introduce some concepts required by this definition.

The P-DeLP system requires a particular argument comparison strategy to deal with conflicting arguments. In Section 3 we have presented a specific criterion that relies on the weights of arguments; these weights are obtained by a specific way of aggregating (using the min function) the priority degrees attached to the defeasible rules in the argument. In our approach, the priority degree associated with a defeasible rule is context-dependent, where the notion of context is understood—in a general sense—as conditions favoring a particular priority criterion. The following definition introduces the concept of priority criterion as an assignment of weights to defeasible rules (recall that the higher is the weight assigned to a defeasible rule, the higher is the priority of that rule).[2]

Definition 4 (Priority criterion). *Let $\mathbb{R}$ be a finite set of rules built from a set of literals $\mathbf{L}$. A priority criterion* prc *is specified by an assignment $\rho_{\mathsf{prc}} : \mathbb{R} \to [0, 1)$ of priority degrees to the rules in $\mathbb{R}$.*

A couple of remarks to simplify notation: (i) we will often identify the criterion identifier prc with its associated assignment of weights ρ_{prc}, and we will simply write $\mathsf{prc}(R)$ instead of $\rho_{\mathsf{prc}}(R)$; and (ii) given a set of (weighted) defeasible rules Δ and a priority criterion prc, we will write Δ_{prc} to denote the set of updated rules $\{(R, \mathsf{prc}(R)) \mid R \in \Delta^-\}$, where $\Delta^- = \{R \mid (R; \omega) \in \Delta$ for some $\omega \in [0, 1]\}$. It is important to note that, if prc assigns a minimal weight to a defeasible rule $R \in \Delta^-$, that is, if $\mathsf{prc}(R) = 0$, then R will play no role at all under the criterion prc. Also note that, since priority criteria are intended to be assigned only to defeasible rules, by definition, it is not allowed to assign a maximal weight 1 to a 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 price and trust prioritizing rules according respectively to the price of what is being offered to the user and how much the user trusts the source of the offer. The following are the corresponding sets of updated rules according to these criteria:

[2]As it is assumed in many non-monotonic reasoning scenarios, there are different ways in which preference or priority degrees for pieces of defeasible knowledge can be specified; for instance, by the knowledge engineer according to their (subjective) priorities, with feedback provided by human users that participate in human-in-the-loop systems, or in the context of multiagent systems by defining a generic procedure for automatically updating an agent's knowledge base when new incoming information is perceived.

$$\Delta_{\mathsf{trust}} = \left\{ \begin{array}{l} (\mathit{insurance}(I, U) \leftarrow \mathit{expIns}(I, U), \mathit{trustIns}(I, U);\ 0.8) \\ (\sim\!\mathit{insurance}(I, U) \leftarrow \mathit{expIns}(I, U);\ 0.3) \\ (\mathit{prefA}(A, U) \leftarrow \mathit{expA}(A), \mathit{trustA}(A, U);\ 0.7) \\ (\sim\!\mathit{prefA}(A, U) \leftarrow \mathit{expA}(A);\ 0.3) \\ (\mathit{prefTAg}(T, U) \leftarrow \mathit{expTAg}(T), \mathit{trusTAg}(T, U);\ 0.9) \\ (\sim\!\mathit{prefTAg}(T, U) \leftarrow \mathit{expTAg}(T);\ 0.5) \\ (\mathit{expIns}(I, U) \leftarrow \mathit{spendIns}(E, U), \mathit{costIns}(C, I), C > E;\ 0.95) \\ [\ldots] \end{array} \right\}$$

$$\Delta_{\mathsf{price}} = \left\{ \begin{array}{l} (\sim\!\mathit{insurance}(I, U) \leftarrow \mathit{expIns}(I, U);\ 0.6) \\ (\mathit{insurance}(I, U) \leftarrow \mathit{expIns}(I, U), \mathit{trustIns}(I, U);\ 0.5) \\ (\sim\!\mathit{prefA}(A, U) \leftarrow \mathit{expA}(A);\ 0.8)) \\ (\mathit{prefA}(A, U) \leftarrow \mathit{expA}(A), \mathit{trustA}(A, U);\ 0.4) \\ (\sim\!\mathit{prefTAg}(T, U) \leftarrow \mathit{expTAg}(T);\ 0.9) \\ (\mathit{prefTAg}(T, U) \leftarrow \mathit{expTAg}(T, D), \mathit{trusTAg}(T, U);\ 0.5) \\ (\mathit{expIns}(I, U) \leftarrow \mathit{spendIns}(E, U), \mathit{costIns}(C, I), C > E;\ 0.95) \\ [\ldots] \end{array} \right\}$$

Consider now the argument comparison strategy defined in Section 3.2. Then, using the priorities specified in Δ_{trust}, the argument $\langle \mathcal{A}_1, (\mathit{insurance}(i1, ana); 0.8) \rangle$ is preferred over the argument $\langle \mathcal{A}_2, (\sim\!\mathit{insurance}(i1, ana); 0.3) \rangle$, where

$$\mathcal{A}_1 = \left\{ \begin{array}{l} (\mathit{insurance}(i1, ana) \leftarrow \mathit{expIns}(i1, ana), \mathit{trustIns}(i1, ana);\ 0.8) \\ (\mathit{expIns}(i1, ana) \leftarrow \mathit{spendIns}(1500, ana), \mathit{costIns}(1700, i1), 1700 > 1500;\ 0.95) \\ (\mathit{trustIns}(i1, ana) \leftarrow \mathit{airline}(a1, ana), \mathit{lostLuggage}(i1);\ 0.85) \end{array} \right\}$$

$$\mathcal{A}_2 = \left\{ \begin{array}{l} (\sim\!\mathit{insurance}(i1, ana) \leftarrow \mathit{expIns}(i1, ana);\ 0.3) \\ (\mathit{expIns}(\mathrm{i1}, ana) \leftarrow \mathit{spendIns}(1500, ana), \mathit{costIns}(1700, i1), 1700 > 1500;\ 0.95) \end{array} \right\}$$

since $\mathcal{A}_1$ provides a greater weight than $\mathcal{A}_2$. However, if the priority criterion price is used, then we get the arguments $\langle \mathcal{A}_1, (\mathit{insurance}(i1, ana); 0.5) \rangle$ and $\langle \mathcal{A}_2, (\sim\!\mathit{insurance}(i1, ana); 0.6) \rangle$, where:

$$\mathcal{A}_1 = \left\{ \begin{array}{l} (\mathit{insurance}(i1, ana) \leftarrow \mathit{expIns}(i1, ana), \mathit{trustIns}(i1, ana);\ 0.5) \\ (\mathit{expIns}(i1, ana) \leftarrow \mathit{spendIns}(1500, ana), \mathit{costIns}(1700, i1), 1700 > 1500;\ 0.95) \\ (\mathit{trustIns}(i1, ana) \leftarrow \mathit{airline}(a1, ana), \mathit{lostLuggage}(i1);\ 0.85) \end{array} \right\}$$

$$\mathcal{A}_2 = \left\{ \begin{array}{l} (\sim\!\mathit{insurance}(i1, ana) \leftarrow \mathit{expIns}(i1, ana);\ 0.6) \\ (\mathit{expIns}(\mathrm{i1}, ana) \leftarrow \mathit{spendIns}(1500, ana), \mathit{costIns}(1700, i1), 1700 > 1500;\ 0.95) \end{array} \right\}$$

and thus $\mathcal{A}_2$ is preferred to $\mathcal{A}_1$. □

One of our goals is to devise a mechanism that dynamically changes preferences depending on the world's current state where the planner system acts. We will formalize this idea in the rest of this section.

Definition 5 (Consistent set of literals/State). *A state of the world Ψ is represented by a consistent set of literals. A set of literals is consistent if it is a non-contradictory set.*

Example 4. Consider a planning domain where a system makes recommendations over tourist destinations considering the user's preferences. The following consistent set of facts can represent a possible state,

$$\Psi_4 = \left\{ \begin{array}{l} \mathit{airline}(a1, ana), \\ \mathit{spendIns}(1500, ana), \\ \mathit{costIns}(1700, i1), \\ \mathit{lostLuggage}(i1), \\ \mathit{covIns}(d1, i1), \\ \mathit{safeDest}(d1) \end{array} \right\}$$

where $\mathit{covIns}(d1, i1)$ expresses that $d1$ is a tourist destination covered by travel insurance policy $i1$, and $\mathit{safeDest}(d1)$ expresses that $d1$ is a safe destination.□

In [28], we introduced a mechanism for changing the argument preference criterion where the use of guards offers a particular way of associating conditions to the selection of a criterion. In this paper, on the other hand, guards allow guiding the choice of a priority order issued from a given context depending on the world's current state instead of selecting a criterion for comparing arguments (as was proposed in [28]).

Definition 6 (Guard). *A guard is a set of literals γ. We will say that a guard is satisfied by a state Ψ when $\gamma \subseteq \Psi$.*

In our approach, defeasible argumentation is used for reasoning over the preconditions to execute actions. To do so in a specific context, the planning system will use a particular priority order over defeasible knowledge obtained after evaluating a conditional expression. Thus, as we will show later, every action will be associated with a conditional expression which will allow for the possibility to consider all priority criteria declared in the system or just some of them. In particular, the use of conditional expressions has been an issue recently addressed in [13] in the context of the DeLP system, where a special kind of conditional expression is used to establish the way in which the system compares arguments. Following the formal structure of the conditional expressions introduced in [13], a particular type of expression adapted to our approach for selecting priority criteria is introduced next.

Definition 7 (Conditional-preference expression). *Let $\mathbf{C}$ be a set of priority criteria over a set of defeasible rules Δ. A conditional-preference expression E over $\mathbf{C}$ is a finite sequence that can be inductively defined as follows:*

- *every priority criterion* $\mathsf{prc} \in \mathbf{C}$ *is a conditional-preference expression over* $\mathbf{C}$*;*

- *if E_1 and E_2 are conditional-preference expressions over* $\mathbf{C}$ *and* γ *is a guard, then* $[\gamma : E_1; E_2]$ *is a conditional-preference expression over* $\mathbf{C}$*;*

- *nothing else is a conditional-preference expression over* $\mathbf{C}$*.*

Intuitively, the conditional-preference expression E in the above definition can be understood as follows: if E is a priority criterion prc, then the priority assignment corresponding to this criterion is applied over the rules of Δ, otherwise if E is $[\gamma : E_1; E_2]$ and γ is satisfied by 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. This intuitive idea is captured by the function eval defined below.

Definition 8 (Evaluating function). *Let* $\mathbf{L}$ *be the set of all literals the planner system can use for representing world states,* $\mathbf{C}$ *be a set of priority criteria over the set of defeasible rules the system manages, and* $\boldsymbol{E}$ *be the set of all possible conditional-preference expressions built over* $\mathbf{L}$ *and* $\mathbf{C}$ *(hence,* $\mathbf{C} \subseteq \boldsymbol{E}$*). Then, the function* $\mathsf{eval} : \boldsymbol{E} \times 2^{\mathbf{L}} \longrightarrow \mathbf{C}$ *for evaluating conditional-preference expressions is defined as:*

$$\mathsf{eval}(E, \Psi) = \begin{cases} \mathsf{prc}, \;\; \textit{if } \; E = \mathsf{prc}, \\[2ex] \mathsf{eval}(E_1, \Psi), \;\; \textit{if } \; E = [\gamma : E_1; E_2] \textit{ and } \gamma \textit{ is satisfied by } \Psi, \\[2ex] \mathsf{eval}(E_2, \Psi), \;\; \textit{if } \; E = [\gamma : E_1; E_2] \textit{ and } \gamma \textit{ is not satisfied by } \Psi. \end{cases}$$

Note that, even when no guard is satisfied, the result of evaluating a conditional-preference expression will always be a priority criterion.

Example 5. Consider the priority criteria price and trust defined in Example 3. Using these criteria, the following are examples of conditional-preference expressions:

— $E_1 = [\{\textit{expensiveDest}(D)\} : \mathsf{trust}; [\{\textit{safeDest}(D), \textit{topDest}(D)\} : \mathsf{price}; \mathsf{trust}]]$

— $E_2 = \mathsf{price}$

Expression E_1 is to be interpreted as follows: "if $\textit{expensiveDest}(D)$ is present in the system's state, then use the trust-based priority, otherwise $[\{\textit{safeDest}(D), \textit{topDest}(D)\} : \mathsf{price}; \mathsf{trust}]$ should be evaluated". Consider the state Ψ_4 introduced in Example 4 and the tourist destination $d1$. For E_1, the guards $\{\textit{expensiveDest}(d1)\}$ and $\{\textit{safeDest}(d1), \textit{topDest}(d1)\}$ are not satisfied by the state Ψ_4. Thus, the result $\mathsf{eval}(E, \Psi_4)$ of evaluating E_1 in the state Ψ_4 is the priority criterion trust. □

The computational mechanisms presented here will allow us to formalize in the next section a planning approach capable of handling preferences based on the use of conditional-preference expressions.

5. Epistemic Planning

We now introduce an epistemic planning formalism that combines actions with argumentative reasoning where contextualized priorities expressing user preferences can be considered. A solution to a planning problem may use a different priority order over defeasible knowledge for each action executed in our proposal. The selection of this ordering will depend on the result of evaluating a conditional-preference expression, which is associated with the chosen action.

Three elements specify an action A: its preconditions P, its consequences X, and the preferences E under which P will be evaluated. Formally,

Definition 9 (Action). *An action is expressed as a tuple* $\mathsf{A} = \langle \mathsf{X}, \mathsf{P}, E \rangle$*, 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:*

$$\mathsf{X} \xleftarrow{(\mathsf{A},E)} \mathsf{P}, \quad \textit{or}$$

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

Intuitively, in a given context, an action A specifies that "*if all literals in* P *representing the preconditions of* A *are warranted under the preferences defined by the criterion* prc *obtained from* E*, then after executing* A *the literals of* X *will be added to the state* Ψ". Later, we will define when an action is applicable and the result of its execution.

Example 6. Consider the application domain presented in Example 4 and the conditional-preference expressions

$$E_1 = [\{\mathit{expensiveDest}(D)\} : \mathsf{trust}; [\{\mathit{safeDest}(D), \mathit{topDest}(D)\} : \mathsf{price}; \mathsf{trust}]], \ \textit{and}$$

$$E_2 = \mathsf{price}$$

of Example 5. Suppose the system has the following actions concerning a tourist destination. The action $\mathsf{recDest}$ allows to recommend a place whenever there exists an available travel insurance policy for the user.

$$\mathbf{A}_6 = \left\{ \begin{array}{l} \{\mathit{airline}(A, U)\} \xleftarrow{(\mathsf{recAirline}(A,U),E_1)} \{\mathit{avAir}(A), \mathit{prefA}(A, U)\} \\ \{\mathit{travelAg}(T, U)\} \xleftarrow{(\mathsf{recTravelAg}(T,U),E_2)} \{\mathit{avTAg}(T), \mathit{prefTAg}(T, U)\} \\ \{\mathit{tDest}(D1, U)\} \xleftarrow{(\mathsf{recDest}(D,U),E_1)} \{\mathit{covIns}(D, I), \mathit{insurance}(I, U)\} \end{array} \right\} \quad \square$$

As we have mentioned, our formalism allows the introduction of the knowledge to be used to reason about the actions and offer a set of actions that will be available for modifying the world. We formalize the notion of preference-based planning domain as follows:

Definition 10 (Planning domain). *A preference-based planning domain is a triple* $(\Delta, \mathbf{C}, \mathsf{\mathbf{A}})$ *where:*

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

Example 7. Consider the set of defeasible rules Δ_1 defined in Example 1, the set of criteria $\mathbf{C}_3 = \{\mathsf{price}, \mathsf{trust}\}$ of Example 3, and the set of actions $\mathsf{\mathbf{A}}_6$ presented in Example 6. The triple $(\Delta_1, \mathbf{C}_3, \mathsf{\mathbf{A}}_6)$ defines the planing domain for the recommender system introduced in Example 4. □

The P-DeLP argumentation formalism described above allows a system to represent domain knowledge with defeasible rules, as defined in Definition 10. These domain rules together with the set of literals describing the current state of the world Ψ define a P-DeLP program (Ψ^*, Δ), where $\Psi^* = \{(L, 1) \mid L \in \Psi\}$, upon which the planner system can perform defeasible reasoning, *e.g.,* about whether preconditions of a given action are warranted. For simplicity in the formalizations and algorithms proposed in this paper, we will consider restricted forms of P-DeLP programs of the form (Ψ^*, Δ) without strict rules. Nevertheless, it is important note that one can have non-attackable defeasible rules that can exactly capture the behavior of strict rules.

In a given planning domain $(\Delta, \mathbf{C}, \mathsf{\mathbf{A}})$, the condition that must be satisfied before an action $\mathsf{A} = \langle \mathsf{X}, \mathsf{P}, E\rangle$ can be executed consists of warranting the literals of the set P. The warrant of these literals will depend on the priority criterion used. As we propose to use a priority selection mechanism based on the current world state, it is possible that a particular action can be associated with different contextual priorities in different circumstances. We will denote by $\mathsf{warrL}(\Psi, \Delta)$ the set of literals warranted by the program (Ψ^*, Δ), that is,

$$\mathsf{warrL}(\Psi, \Delta) = \{L \mid (L, \omega) \text{ is warranted by } (\Psi^*, \Delta) \text{ for some } \omega > 0\}$$

Definition 11 (Applicable action). *Let* $(\Delta, \mathbf{C}, \mathsf{\mathbf{A}})$ *be a planning domain, and* Ψ *a state. Let* $\mathsf{A} = \langle \mathsf{X}, \mathsf{P}, E\rangle$ *be an action in* $\mathsf{\mathbf{A}}$ *and* prc *the priority criterion obtained from* $\mathsf{eval}(E, \Psi)$*. The action* A *is applicable in* Ψ *according to preferences defined by* prc *when for every precondition* $P_i \in \mathsf{P}$*, it holds* $P_i \in \mathsf{warrL}(\Psi, \Delta_{\mathsf{prc}})$*.*

After an applicable action $\mathsf{A} = \langle \mathsf{X}, \mathsf{P}, E\rangle$ is executed in a state Ψ, the state itself is consistently modified with each effect of X after removing any possible conflict as described below. We will denote by Ψ^{A} the new state resulting from executing the action A.

Definition 12 (State update function). *Let* $\mathbf{L}$ *be the set of all literals that a system can use for representing world states. Let* $(\Delta, \mathbf{C}, \mathsf{\mathbf{A}})$ *be a planning* *domain, Ψ a state, and $\langle \mathsf{X}, \mathsf{P}, E\rangle$ an action in $\mathbf{A}$ applicable in Ψ. Then, we denote the resulting state after executing A in the state Ψ with*

$$\mathsf{execA}(\mathsf{A}, \Psi) = \Psi^{\mathsf{A}} = (\Psi \backslash \overline{\mathsf{X}}) \cup \mathsf{X},$$

where $\overline{\mathsf{X}}$ is the set of the complemented literals in X.

It is easy to check that the resulting state Ψ^{A} is a valid state, *i.e.,* Ψ^{A} is a consistent set of literals.

Example 8. Let us return to the planing domain $(\Delta_1, \mathbf{C}_3, \mathbf{A}_6)$ for the recommender system specialized in tourist destinations introduced in Example 7 and the state Ψ_4 presented in Example 4, where:

$$\Psi_4 = \left\{ \begin{array}{l} \mathit{airline}(a1, \mathit{ana}), \\ \mathit{spendIns}(1500, \mathit{ana}), \\ \mathit{costIns}(1700, i1), \\ \mathit{lostLuggage}(i1), \\ \mathit{covIns}(d1, i1), \\ \mathit{safeDest}(d1) \end{array} \right\}$$

Consider now the action $\mathsf{recDest}(d1, \mathit{ana})$ in $\mathbf{A}_6$ and the priority criterion trust. This action is applicable in Ψ_4 according to the priorities defined by trust because one of its preconditions $\mathit{covIns}(d1, i1)$ belongs to Ψ_4 and its other precondition $\mathit{insurance}(i1, \mathit{ana})$ belongs to $\mathsf{warrL}(\Psi_4, \Delta_{\mathsf{trust}})$ since there exists a non-defeated argument for it $\langle \mathcal{A}_1, (\mathit{insurance}(i1, \mathit{ana}); 0.8)\rangle$ where:

$$\mathcal{A}_1 = \left\{ \begin{array}{l} (\mathit{insurance}(i1, \mathit{ana}) \leftarrow \mathit{expIns}(i1, \mathit{ana}), \mathit{trustIns}(i1, \mathit{ana});\ 0.8) \\ (\mathit{expIns}(i1, \mathit{ana}) \leftarrow \mathit{spendIns}(1500, \mathit{ana}), \mathit{costIns}(1700, i1), 1700 > 1500;\ 0.95) \\ (\mathit{trustIns}(i1, \mathit{ana}) \leftarrow \mathit{airline}(a1, \mathit{ana}), \mathit{lostLuggage}(i1);\ 0.85) \end{array} \right\}$$

The resulting state of executing the action $\mathsf{recDest}(d1, \mathit{ana})$ in the state Ψ_4 is then the following:

$$\Psi^{\mathsf{recDest}(d1, \mathit{ana})} = (\Psi_4 \backslash \overline{\mathsf{X}}) \cup \mathsf{X} = \left\{ \begin{array}{l} \mathit{airline}(a1, \mathit{ana}) \\ \mathit{spendIns}(1500, \mathit{ana}) \\ \mathit{costIns}(1700, i1) \\ \mathit{lostLuggage}(i1) \\ \mathit{covIns}(d1, i1) \\ \mathit{safeDest}(d1) \\ \mathit{tDest}(d1, \mathit{ana}) \end{array} \right\}$$

where $\mathsf{X} = \{\mathit{tDest}(d1, \mathit{ana})\}$. □

Since the execution of an applicable action leads to a new state, another action could be applicable at this new state, and so on. Therefore, this observation naturally leads to the notion of an *applicable sequence of actions.*

Definition 13 (Applicable sequence). *Given a planning domain* $(\Delta, \mathbf{C}, \mathbf{A})$ *and a state* Ψ*, let* $\mathsf{S} = [\mathsf{A}_1, \mathsf{A}_2, \dots, \mathsf{A}_n]$ *be a sequence of actions where each* $\mathsf{A}_i \in \mathbf{A}$*. The sequence* S *will be regarded as an* applicable sequence *of actions at* Ψ *if* (1) A_1 *is applicable at* Ψ*, and* (2) *for 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* $\Psi^{[\mathsf{A}_1, \dots, \mathsf{A}_n]}$ *as a shorthand for* $(^{\cdots}(\Psi^{\mathsf{A}_1})^{\cdots})^{\mathsf{A}_n}$*.*

Again, it is immediate to verify that the resulting state $\Psi^{[\mathsf{A}_1, \mathsf{A}_2, \dots, \mathsf{A}_n]}$ of executing an applicable sequence of actions $\mathsf{S} = [\mathsf{A}_1, \mathsf{A}_2, \dots, \mathsf{A}_n]$ is a consistent set of literals.

Example 9. Consider the planning domain $(\Delta_1, \mathbf{C}_3, \mathbf{A}_6)$ introduced in Example 7 and recall the set of actions

$$\mathbf{A}_6 = \left\{ \begin{array}{l} \{airline(A, U)\} \xleftarrow{(\mathsf{recAirline}(A,U), E_1)} \{avAir(A), prefA(A, U)\} \\ \{travelAg(T, U)\} \xleftarrow{(\mathsf{recTravelAg}(T,U), E_2)} \{avTAg(T), prefTAg(T, U)\} \\ \{tDest(D1, U)\} \xleftarrow{(\mathsf{recDest}(D,U), E_1)} \{covIns(D, I), insurance(I, U)\} \end{array} \right\}$$

where $E_1 = [\{expensiveDest(D)\} : \mathsf{trust}; [\{safeDest(D), topDest(D)\} : \mathsf{price}; \mathsf{trust}]]$ and $E_2 = \mathsf{price}$. Let Ψ_9 be the following state:

$$\Psi_9 = \left\{ \begin{array}{l} spendIns(1500, ana), costIns(1700, i1), lostLuggage(i1), \\ avAir(a1), trustA(a1, juan), trust(ana, juan), \\ topRatingA(a1), safeDest(d1), covIns(d1, i1) \end{array} \right\}$$

Different sequences of actions are analyzed in order to determine whether they are applicable at Ψ_9 below.

- Consider the action sequence

  $$\mathsf{S}_1 = [\mathsf{recDest}(d1, ana)].$$

  In this case, the working priority criterion is $\mathsf{eval}(E_1, \Psi_9) = \mathsf{trust}$ and it turns out that the action $\mathsf{recDest}(d1, ana)$ is not applicable in Ψ_9 because the precondition $insurance(i1, ana)$ does not belong to $\mathsf{warrL}(\Psi_9, \Delta_{\mathsf{trust}})$. Then, S_1 is not a feasible plan since it is not a sequence of applicable actions at Ψ_9.

- Consider the sequence

  $$\mathsf{S}_2 = [\mathsf{recTravelAg}(t1, ana), \mathsf{recDest}(d1, ana)].$$

Given preferences defined by the criterion price, obtained after evaluating the expression E_2 from Ψ_9, the action $\mathsf{recTravelAg}(t1, ana)$ is not applicable from Ψ_9 since an agency preferred by ana cannot be found from $(\Psi_9, \Delta_{\mathsf{price}})$. Thus, the sequence S_2 is not applicable from Ψ_9.

- Now, let us look at the sequence

$$\mathsf{S}_3 = [\mathsf{recAirline}(a1, ana), \mathsf{recDest}(d1, ana)].$$

Using the priorities Δ_{trust} defined by the criterion trust obtained after evaluating E_1 from Ψ_9, $a1$ is an available airline ($avAir(a1)$) and preferred by ana ($prefA(a1, ana)$). Thus, the action $\mathsf{recAirline}(a1, ana)$ is applicable. The result of executing $\mathsf{recAirline}(a1, ana)$ is:

$$\Psi_9^{[\mathsf{recAirline}(a1,\, ana)]} = \left\{ \begin{array}{l} spendIns(1500, ana), costIns(1700, i1), avAir(a1), \\ lostLuggage(i1), trustA(a1, juan), trust(ana, juan), \\ topRatingA(a1), safeDest(d1), airline(a1, ana), \\ covIns(d1, i1) \end{array} \right\}$$

The next action $\mathsf{recDest}(d1, ana)$ is also applicable at $\Psi_9^{[\mathsf{recAirline}(a1,ana)]}$, and observe that

$$covIns(d1, i1) \in \Psi_9^{[\mathsf{recAirline}(a1,ana)]}$$

and

$$insurance(i1, ana) \in \mathsf{warrL}(\Psi_9^{[\mathsf{recAirline}(a1,\, ana)]}, \Delta_{\mathsf{trust}})$$

since there exists the non-defeated argument $\langle \mathcal{A}_1, (insurance(i1, ana); 0.8)\rangle$ for $(insurance(i1, ana), 0.8)$, where

$$\mathcal{A}_1 = \left\{ \begin{array}{l} (insurance(i1,\, ana) \leftarrow expIns(i1,\, ana), trustIns(i1,\, ana);\ 0.8) \\ (expIns(i1,\, ana) \leftarrow spendIns(1500,\, ana), costIns(1700,\, i1), 1700 > 1500;\ 0.95) \\ (trustIns(i1,\, ana) \leftarrow airline(a1,\, ana), lostLuggage(i1);\ 0.85) \end{array} \right\}$$

The resulting state after executing $\mathsf{recDest}(d1, ana)$ from $\Psi_9^{[\mathsf{recAirline}(a1,ana)]}$ is the following:

$$\Psi_9^{\mathsf{S}_3} = \left\{ \begin{array}{l} spendIns(1500, ana), costIns(1700, i1), lostLuggage(i1), avAir(a1) \\ trust(ana, juan), topRatingA(a1), safeDest(d1), trustA(a1, juan), \\ airline(a1, ana), tDest(d1, ana) \end{array} \right\}$$

and S_3 is a sequence of applicable actions at Ψ_9. $\square$

The main aim of classical planning is to find a sequence of actions that, starting from an initial state, leads to a state where a given goal is satisfied. Next, we will formally define the concept of preference-based planning problem for a given domain.

Definition 14 (Preference-based planning problem). *Let* $(\Delta, \mathbf{C}, \mathsf{\mathbf{A}})$ *be a planning domain. A preference-based planning problem over* $(\Delta, \mathbf{C}, \mathsf{\mathbf{A}})$ *is a tuple* $(\Psi, \Delta, \mathbf{C}, \mathsf{\mathbf{A}}, \mathbb{G})$*, where:*

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

— Δ *is a set of defeasible rules,*

— $\mathbf{C}$ *is a set of priority criteria,*

— $\mathsf{\mathbf{A}} = \{\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n\}$ *is a set of actions, and*

— $\mathbb{G}$ *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 will lead to a state that satisfies the conditions in $\mathbb{G}$. Formally:

Definition 15 (Solution plan to a preference-based planning problem). *Let* $T = (\Psi, \Delta, \mathbf{C}, \mathsf{\mathbf{A}}, \mathbb{G})$ *be a preference-based planning problem. We will say that a sequence of actions* $\mathsf{S} = [\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ *is a solution for* T *if:*

(*i*) S *is applicable at* Ψ*, and*

(*ii*) *Each literal* $L \in \mathbb{G}$ *is warranted by the updated program* $((\Psi^{\mathsf{S}})^*, \Delta')$*, where* Δ' *denotes the set of updated defeasible rules resulting from the application of some priority criterion; that is,* $\mathbb{G} \subseteq \mathsf{warrL}(\Psi^{\mathsf{S}}, \Delta')$.

Observe that in the above definition, if $\mathsf{S} = [\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ is applicable at Ψ and $\mathbb{G} \subseteq \Psi^{\mathsf{S}}$, each $L \in \mathbb{G}$ is automatically warranted in the program $(\Psi^{\mathsf{S}}, \Delta')$ since the argument $\langle \emptyset, (L; 1) \rangle$ is undefeated. The intuitions behind this definition also contemplate situations where goals are not explicitly achieved in the final state, but they are conclusions derived from program rules. The main rationale for considering these situations is that there could be effects produced by the interaction of actions with the environment or with other actions, which it does not seem to be a feasible alternative to consider all possible effects in an action's definition. Consider, for instance, the sequence of applicable actions S_3 at Ψ_9 presented in Example 9, and assume that $\mathbb{G} = \{tDest(d1, ana), insurance(i1, ana)\}$. In this case, it turns out that $tDest(d1, ana), insurance(i1, ana) \in \mathsf{warrL}(\Psi_9^{[\mathsf{recAirline}(a1,\, ana),\, \mathsf{recDest}(d1,\, ana)]}, \Delta_{\mathsf{trust}})$, but $insurance(i1, ana) \notin \Psi_9^{\mathsf{S}_3}$, *i.e.,* it is not computed as an action's effect. This capacity is especially useful in dynamic domains where it is not always possible to have all the information necessary to achieve the planning system's goals in advance.

Example 10. Given the conditional-preference expression E_2 presented in Example 5, the following preference-based planning problem $(\Psi_9, \Delta_1, \mathbf{C}_3, \mathsf{A}_6, \mathbb{G}_{10})$ can be formulated, where

— Ψ_9 is the state presented in Example 9,

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

— $\mathbb{G}_{10} = \{\mathit{tDest}(d1, ana), \mathit{airline}(a1, ana)\}$

A possible solution for this planning problem is the plan

$$\mathsf{S}_3 = [\mathsf{recAirline}(a1, ana), \mathsf{recDest}(d1, ana)]$$

since

— S_3 is a sequence of applicable actions at Ψ_9, and

— $\mathbb{G}_{10} \subseteq \Psi^{\mathsf{S}_3}$.

An applicable sequence of actions may exist that does not represent a solution to the planning problem at hand; for instance, if we consider a planning problem slightly different from the previous one, such as the following: $T' = (\Psi_9, \Delta_1, \mathbf{C}_3, \mathsf{A}_6, \mathbb{G}')$ where

$$\mathbb{G}' = \{\mathit{tDest}(d1, ana), \mathit{travelAg}(t1, ana)\},$$

it turns out that S_3 is still applicable at Ψ_9, but it is no longer a solution for T'.
□

So far, we have presented a planning formalism that integrates preferences into the construction of plans. In particular, the proposed approach provides the possibility of expressing contextual preferences under which a specific action's preconditions should be evaluated. To encode these preferences, we have introduced conditional priority expressions, allowing the user to specify possibly different priority criteria depending on the state of the world. This feature of our formalism is intended to help produce satisfactory solution plans concerning the user's needs and preferences, in the sense of favoring their confidence in the obtained plans.

In the next section, we introduce an extension of the APOP algorithm presented in [7] in order to properly deal with the conditional expressions proposed in this work. We first analyze different types of interferences that may arise when conditional expressions are used, and then, in Section 7, go on to analyze computational complexity issues associated with our approach.

6. Argumentation-based Partial Order Planning with Contextual Preferences (P-APOP)

The formalism described above defines when actions are applicable and how to compute their effects, but it does not describe *how* to construct a plan for achieving a planner system's goals. In [7], a Partial Order Planning algorithm for Defeasible Logic Programming, called APOP, is proposed to build plans. This section will introduce an extension of this algorithm to consider conditional-preference expressions during the planning process.

The basic idea behind the APOP algorithm is to search through a plan space, beginning from the system's goals up to a given initial state, to generate 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 by warranting them through the argumentation process. This initial plan is then incrementally completed with new steps until all the steps' preconditions are warranted. Intuitively, this process generates a new partial plan whenever a new step is considered. As usual in the partial-order planning paradigm [29], a plan in APOP includes all actions that need to be taken and specifies an ordering between actions only when necessary; in contrast, in total-order planning, a plan specifies a linear order on the actions to be taken. In [7], two types of steps are identified: *action steps* for representing the execution of an action, and *argument steps* for expressing arguments used in the plan to support the preconditions of some action step. Unlike actions, arguments will not only be used to support some step of a plan, but they will also be added as interfering or supporting arguments in the plan.

If only actions are used in a plan, then only one type of interference (threat) can arise in a plan under construction. This interference appears when a new action added in the plan deletes a literal satisfying a precondition already solved by other action steps. Nevertheless, when involving actions and arguments to construct plans, other types of interferences can appear, which should be addressed appropriately to obtain a valid plan. In [7], the authors identify different types of interferences that could arise in argumentation-based planning and propose methods to resolve each of them. When conditional expressions are used, an action might have interferences with the guards appearing in these expressions. In Section 6.3, we will introduce an extension of the APOP algorithm that detects these new threats and attempts to resolve them. However, before addressing this, we first introduce some basic terminology and graphical representation used in the rest of the paper.

6.1. Basic terminology and graphical representation

In this section, our main interest is to study a new type of threat that arises when an action's effects interfere with guards of conditional-preference expressions already evaluated. Therefore, we need to detect under which circumstances they could happen to decide how to address them. To facilitate this task, and before showing with an illustrative example how a complete plan is constructed in P-APOP, we will introduce the notion of *selected path structure* below.

Let us recall that in the epistemic planning formalism described in Section 5, an action is applicable in a current state Ψ if every precondition of the action is warranted under the rule priority order obtained from the evaluation of the conditional expression E associated with the action. In this process, the satisfaction of the guards is checked against Ψ. It is interesting to observe that, a priori, for any criterion in E, there exists a finite sequence of guards whose satisfaction or unsatisfaction support the selection of such a criterion. Of course, different sequences of guards can possibly lead to different criteria. Nevertheless, the evaluation of an expression E in a given state Ψ, resulting in a criterion $\mathsf{eval}(E, \Psi)$ (see Definition 8), reflects the fact that there exists only one sequence of guards whose satisfaction status by the state leads to that criterion. For instance, considering the expression E_1 and state Ψ_9 of Example 9, $\mathsf{eval}(E_1, \Psi_9)$ returns the criterion trust, and its associated sequence of guards is $\mathcal{G}_2 = [\{\mathit{expensiveDest}(D)\}, \{\mathit{safeDest}(D),\mathit{topDest}(D)\}]$, where guards $\{\mathit{expensiveDest}(D)\}$ and $\{\mathit{safeDest}(D),\mathit{topDest}(D)\}$ are not satisfied by Ψ_9.

The following definition is introduced to identify which guards are satisfied and which ones are not by a given state when the conditional-preference expression associated with a specific action is evaluated. The sequence of guards supporting the selection of a priority criterion is considered below.

Definition 16 (Selected path structure). *Let E be a conditional-preference expression, Ψ a state, and prc the priority criterion obtained by the evaluation of E in Ψ, that is, $\mathsf{prc} = \mathsf{eval}(E, \Psi)$. Let $\mathcal{G} = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ be the sequence of guards appearing in E whose evaluation leads to obtain prc. The selected path structure for prc extracted from $\mathcal{G}$ is the triple $\Gamma = (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E}$ where:*

— $\mathcal{G}^+ = \{\gamma \in \mathcal{G} \mid \gamma$ *is satisfied by* $\Psi\}$*, and*

— $\mathcal{G}^- = \{\gamma \in \mathcal{G} \mid \gamma$ *is not satisfied by* $\Psi\}$*.*

Example 11. Consider the conditional-preference expressions

$$E_1 = [\{\mathit{expensiveDest}(D)\} : \mathsf{trust}; [\{\mathit{safeDest}(D), \mathit{topDest}(D)\} : \mathsf{price}; \mathsf{trust}]]$$

$$E_2 = \mathsf{price}$$

of Example 5. Given the state Ψ_9 of Example 9, the following two selected path structures can be obtained:

- $\Gamma_1 = (\{\}, \{\{\mathit{expensiveDest}(d1)\}, \{\mathit{safeDest}(d1), \mathit{topDest}(d1)\}\}, \mathsf{trust})_{\Psi_9,E_1}$
- $\Gamma_2 = (\{\}, \{\}, \mathsf{price})_{\Psi_9,E_2}$ □

The definition above introduces a structure later used to identify when an action step threatens a guard in a selected path structure. Next, we will use our example domain of tourist destination recommendation for illustrative purposes to show how APOP can be extended to handle the conditional preference expressions presented before.

Figure 1 illustrates a sequence of partial plans and how a complete plan using actions and arguments for Example 10 is obtained. The preconditions of the finish step correspond to the system's goals $\mathbb{G}_{10}$ and the effects of the start step encode the initial state Ψ_9. For simplicity, we have replaced the literals in $\Psi_9 = \{\mathit{spendIns}(1500, \mathit{ana}), \mathit{costIns}(1700, i1), \mathit{lostLuggage}(i1), \mathit{avAir}(a1),$

$trustA(a1, juan), trust(ana, juan), topRatingA(a1), safeDest(d1), covIns(d1, i1)\}$ by the symbols $sI, cL, lH, aA, tA, t, tR, sD$, and aI, respectively.

In Figure 1, following the graphical representation proposed in [7] to show a complete plan with arguments and actions, 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 path structure 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* ($\rightarrow$) of the plan, and they are used to link an action step effect 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* ($\succ\!-$) of the plan. The *ordering constraints* ($\prec$) are represented by dashed arrows. These constraints allow establishing an order between steps, whereas causal and support links allow identifying each literal source in a plan. These particular sets are later used by algorithms introduced in Section 6.3.

In Figure 1-(a), the finish action step has two unsatisfied preconditions ($tDest(d1, ana)$, $airline(a1, ana)$). The action recDest is the only one available that can be used to satisfy the precondition with the literal $tDest(d1, ana)$. Thus, recDest is added (Fig. 1-(b)) to the plan by the planning process and its preconditions become subgoals to be achieved. Observe that the precondition $covIns(d1, i1)$ is achieved by the start step; however, none of the available actions achieve $insurance(i1, ana)$. Nevertheless, from the rules Δ_{trust}, it is possible to construct the argument $\langle \mathcal{A}_1, (insurance(i1, ana); 0.8)\rangle$ that supports $(insurance(i1, ana);\ 0.8)$, and $\langle \mathcal{A}_2, ({\sim}insurance(i1, ana); 0.3)\rangle$ that attacks $\langle \mathcal{A}_1, (insurance(i1, ana); 0.8)\rangle$ (for the detailed structure, see Example 8). Then, the argument

$$\langle \mathcal{A}_1, (insurance(i1, ana); 0.8)\rangle$$

is selected since it has a greater weight, and the set of literals appearing in the body of rules conforming $\mathcal{A}_1$ become new subgoals; this situation is depicted in Fig. 1-(c). Now, the literals $lostLuggage(i1)$, $spendIns(1500, ana)$, and $costIns(1700, i1)$ are satisfied by the start step, whereas $airline(a1, ana)$ is the effect of the action recAirline. Thus, a new step recAirline is added and now the preconditions $prefA(a1, ana)$ and $avAir(a1)$ must be satisfied. The literal $avAir(a1)$ is satisfied by the start step and from Δ_{trust} it is possible to construct the argument $\langle \mathcal{A}_3, (prefA(a1, ana); 0.2)\rangle$, that is undefeated, supporting the subgoal $prefA(a1, ana)$. Observe that although there is an attacking argument $\langle \mathcal{A}_4, ({\sim}prefA(a1, ana); 0.3)\rangle$, there is no defeater for $\langle \mathcal{A}_3, (prefA(a1, ana); 0.2)\rangle$. The literals in the base of $\langle \mathcal{A}_3, (prefA(a1, ana); 0.2)\rangle$ are achieved by the start step. Finally, the initial goal $airline(a1, ana)$ is achieved by the action already selected (recAirline) and a plan is formulated. In this particular plan, the same selected path structure Γ_1 was obtained from actions, while trust is in turn the

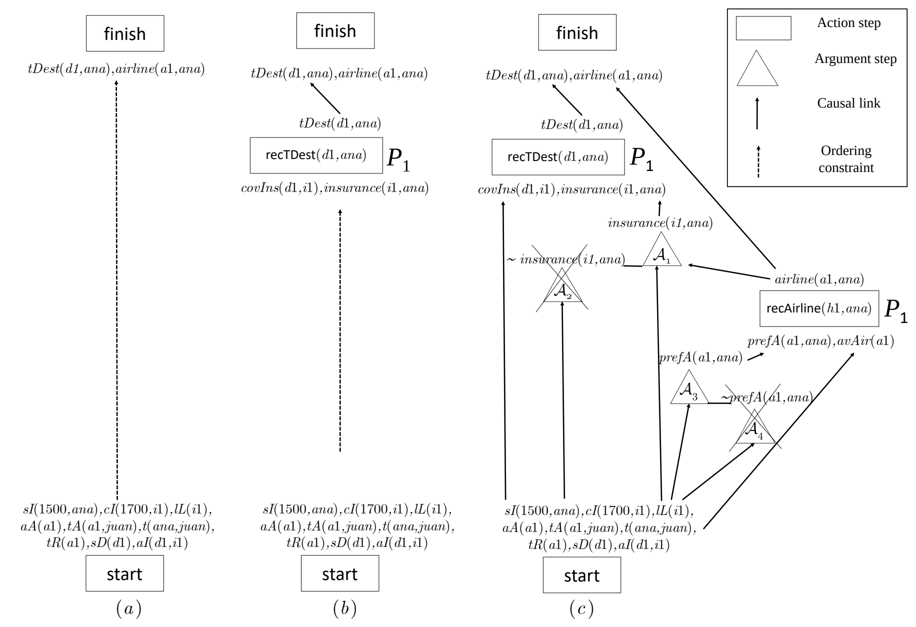


Figure 1: Different partial plans for Example 10.

priority criterion under which such plan is constructed.

In P-APOP, several (potential) arguments can be considered during a plan construction; $\langle \mathcal{A}_1, (insurance(i1, ana); 0.8)\rangle$ and $\langle \mathcal{A}_3, (prefA(a1, ana); 0.2)\rangle$ are two potential arguments. A potential argument is different from the notion used in Section 3 because it is not constructed from a set of facts as usual in the DeLP formalism. It is impossible to know which literals in Ψ are true because they depend on actions that will be chosen later in the planning process. The following definition, presented in [7], is adapted to our proposal to formalize the notion of *potential argument*, that will be used in our P-APOP algorithm in Section 6.3.

Definition 17 (Potential argument). *Let Δ be a finite set of defeasible rules, and $\mathcal{A} \subseteq \Delta$. We say that $\langle \mathcal{A}, (L; \omega)\rangle$ is a potential argument for a literal L with necessity degree $\omega > 0$ if $\langle \mathcal{A}, (L; \omega)\rangle$ is an argument w.r.t. DeLP-program $(Base(\mathcal{A}), \Delta)$, where $Base(\mathcal{A})$ denotes the set of literals that appear in the bodies but not in the heads of the rules in $\mathcal{A}$, i.e., the literals necessary to "activate" the argument allowing to obtain L.*

As mentioned above, $\langle \mathcal{A}_1, (insurance(i1, ana); 0.8)\rangle$ is a potential argument since its existence will depend on the effects of the action recAirline, apart from some literals appearing in the start step. It is important to remark that a potential argument keeps the same structure as an argument as formalized in Definition 1—the difference is that in a "normal" argument $\mathcal{A}$, all the necessary literals are provided by the strict part of the program, *i.e.,* $\Pi \vdash \mathit{Base}(\mathcal{A})$.

In P-APOP, finding a partial-order plan consists in completing a plan by adding steps to achieve subgoals, as illustrated in Figure 1. In this sense, a plan can be seen as a sequence of plan steps, where each step is either an action step or an argument step. More formally, and following descriptions of the P-APOP algorithm (see Algorithm 2), we will regard:

- an action step as a tuple $(\mathsf{N}, \mathsf{X}, \mathsf{P}, \Gamma)$, where N is the step name, $\langle \mathsf{X}, \mathsf{P}, E \rangle$ is the action attached to N, and Γ is the selected path structure associated with E.

- an argument step denoted as $(\mathsf{N}, \langle \mathcal{A}, (L; \omega) \rangle, \Lambda, \mathsf{prc})$, where N is the step name, $\langle \mathcal{A}, (L; \omega) \rangle$ is a potential argument from Δ_{prc}, Λ is a argumentation line associated with N.

As a final remark, note that our proposal 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 to a preference-based planning problem, it is necessary to firstly establish a correspondence between partially-ordered plans and totally-ordered plans, as usual in the partial-order planning paradigm. To achieve this, we can simply apply a topological sorting algorithm to derive a total-order solution. Given a totally-ordered sequence of action steps $\mathit{Seq} = [(\mathsf{start}, \_, \_, \_), (\mathsf{N}_1, \_, \_, \_), \ldots, (\mathsf{N}_n, \_, \_, \_), (\mathsf{finish}, \_, \_, \_)]$[3] derived from a particular partial plan, where each $\mathsf{N}_i \prec \mathsf{N}_j$ $(1 \leq i < j \leq n)$ is consistent with the ordering constraints of the corresponding plan, we will denote by $\mathit{Plan}(\mathit{Seq}) = [\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ 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 $\mathit{Plan}(\mathit{Seq})$ because they do not correspond to the execution of an action—they are only required to represent the initial state and goals of the problem. Finally, we will say that a partial-order plan is a solution to a preference-based planning problem T when the sequence of actions $\mathit{Plan}(\mathit{Seq}) = [\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ extracted from a linearization *Seq* of such a plan is a solution to T. The implementation details such as backtracking over those choice points that lead to failure, necessary for guaranteeing planner completeness, are considered in Section 6.3.

6.2. Interferences with Guards

In APOP [6, 7], when a plan is being constructed, three types of threats involving actions and arguments must be checked (see Figure 2):

Fig. 2-(a) *action-action*: Let $\mathsf{N}_i \rightarrow \mathsf{N}_j$ be a causal link between two action steps; a precondition L of N_j is threatened by an action step N_k if the complemented literal $\overline{L}$ is an effect of N_k.

[3] We use "_" to denote anonymous variables, i.e., those for which value bindings are not relevant.

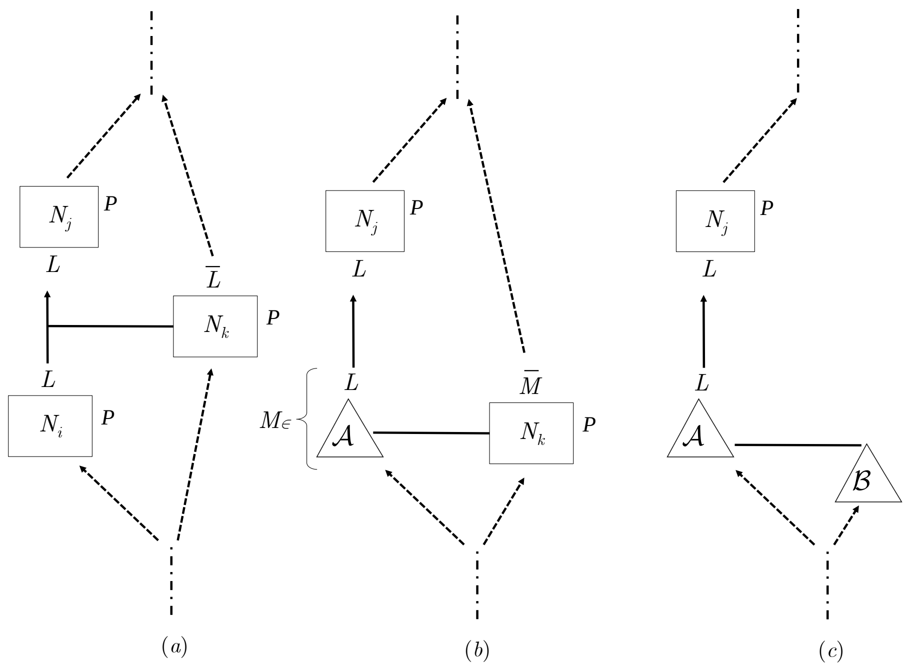


Figure 2: Type of threats in APOP.

Fig. 2-(b) *action-argument*: Let $\langle \mathcal{A}, (L;\alpha)\rangle$ be an argument supporting a precondition of an action step N_j; then, an action step N_k threatens the argument $\langle \mathcal{A}, (L;\alpha)\rangle$ if an effect of N_k negates any literal present in the set of all literals that appear as bodies of rules in the argument $\langle \mathcal{A}, (L;\alpha)\rangle$. The step N_k comes before $\langle \mathcal{A}, (L;\alpha)\rangle$.

Fig. 2-(c) *argument-argument*: Let $\langle \mathcal{A}, (L;\alpha)\rangle$ be an argument added to a plan to support the precondition of an action step N_j; 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.

Different threat resolution methods may be applied for each 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. The detailed study of these threats, and methods to solve them, is out of the scope of this article; the related issues have been thoroughly studied in [7][4].

[4]In particular, the reader is referred to algorithm Resolve_Threats in that work.

Here, we will consider a new type of threat involving guards appearing when extending APOP with conditional expressions proposed in our approach. Let us consider the following case. Given the selected path structure

$$\Gamma_1 = (\{\}, \{\{\mathit{expensiveDest}(d1)\}, \{\mathit{safeDest}(d1), \mathit{topDest}(d1)\}\}, \mathsf{trust})_{\Psi_9, E_1}$$

suppose for instance that $\mathit{topDest}(d1)$ is an effect of action $\mathsf{recAirline}(a1, ana)$. In such a case, this action would interfere with the guard's literals of P_1. If $\mathsf{recAirline}(a1, ana)$ adds $\mathit{topDest}(d1)$ to the current state before $\mathsf{recDest}(d1, ana)$ is executed, the guard $\{\mathit{safeDest}(D), \mathit{topDest}(D)\}$ will be satisfied at the moment E_1 is evaluated; in contrast with P_1, $\{\mathit{safeDest}(D), \mathit{topDest}(D)\}$ will be now present in the set $\mathcal{G}^+$. The evaluation procedure would return the criterion price, and consequently the resulting plan would not be valid. Next, we will show some cases of interferences arising when conditional expressions are used.

Given a selected path structure $(\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E}$, an action step might interfere with the guard's literals of $\mathcal{G}^+$ and $\mathcal{G}^-$. Figures 3-(a), 3-(b), and 3-(c) show three different situations where the action step N_i threatens the guards of $(\{\{d\}\}, \{\}, \mathsf{prc})_{\Psi,E}$ because the effect $\sim d$ negates a literal present in a guard of $\mathcal{G}^+$. If N_i makes $\sim d$ true before N_j, the literal d is not present in the system's state Ψ at the moment the expression E is evaluated. As this can lead to selecting a different priority criterion, N_j could not be executed since a warranted precondition can become non-warranted under the new selected criterion. In addition to these situations, Figure 3-(d) shows yet another different interference situation where N_i makes d true before N_j. Then, when E is evaluated, the literal d will be present in the considered state and the selected path structure associated with the action N_i possibly changes. Note that if we consider threats that involve guards in $\mathcal{G}^+$, its associated path structure will be always affected because the satisfaction state of such guards becomes non-satisfied. However, the situation is different when we consider an action step that interferes with guards of $\mathcal{G}^-$.

Observe that, even though the action N_i can threaten a guard of a path structure, the selected priority criterion may not be affected. Consider again the example in Figure 3-(d), and assume that $(\{\}, \{\{d, a\}\}, \mathsf{prc})_{\Psi,E}$ is the selected path structure associated with N_j. Also, it will be assumed that both literals in $\{d, a\}$ are not satisfied by the system's state. In such a case, although N_i threatens the guard $\{d, a\}$, the selected path structure is not affected since $\{d, a\}$ keeps being non-satisfied. In this particular case, the only possibility to modify the path structure is making both "d" and "a" true before N_j.

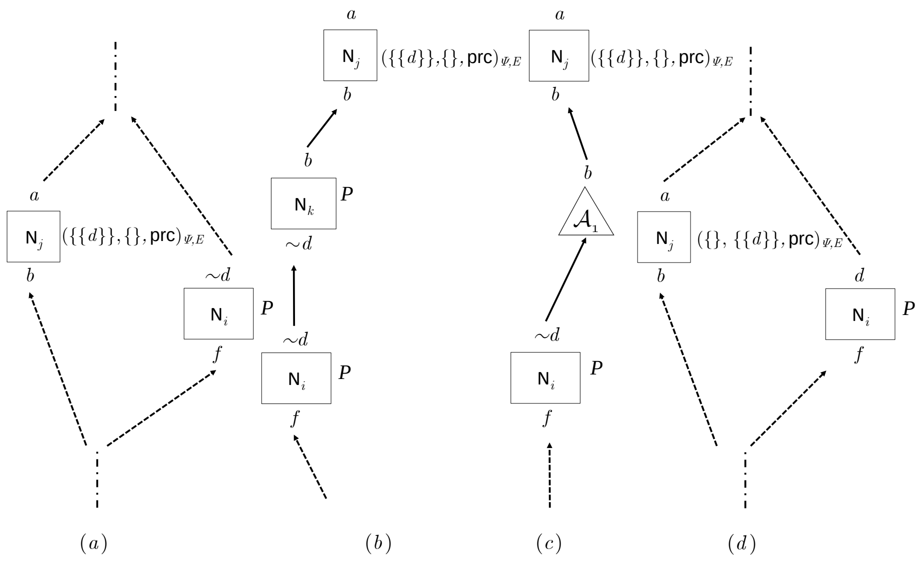


Figure 3: Guard-action threats.

Here, our main interest is to address threats where a selected path structure is affected after a new action is added to the plan. As mentioned above, this type of threats is important because they can lead to select a new criterion that can change in turn the warrant status of an action's precondition. In our proposal, both the interferences with the guards of $\mathcal{G}^+$ and $\mathcal{G}^-$ will be resolved by the same resolution methods independently of the threat type. We consider two possible alternatives to resolve these threats:

- *Promotion*: To add an ordering constraint $(\mathsf{N}_j \prec \mathsf{N}_i)$ to force N_i to come after N_j (see Fig. 4(b)).
- *Disabling a literal*: To add the literal L to N_j as a new precondition to forbid the existence of $\overline{L}$ in the state upon which preference-conditional expression associated with N_j will be evaluated. The step chosen to support L must be an action step to come after N_i because it is necessary to avoid the existence of $\overline{L}$ (see Fig. 4(c)).

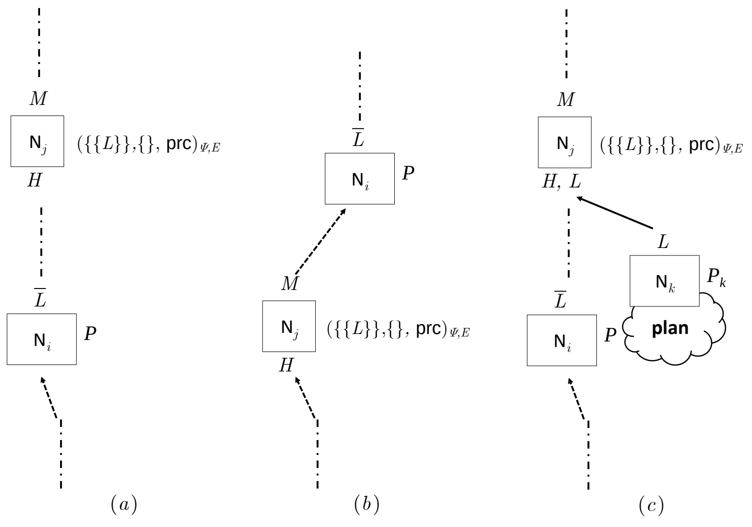


Figure 4: Solutions to guard-action threats.

According to the observations made in the examples above, it is important to note that a threat to a guard literal in $\mathcal{G}^-$ is not enough to change the status of the guard from non-satisfied to satisfied. Indeed, in the case an interference with a guard $\gamma \in \mathcal{G}^-$ occurs, it is necessary to check whether *all* non-satisfied literals of γ by the current state Ψ are effectively threatened by action steps. Only in that case may the status of γ change, and accordingly the path structure associated with N_j may be affected as well. In particular, this situation is considered in the algorithm presented in Section 6.3.

Several conclusions can be drawn from the issues addressed in this section. In general terms, actions could interfere with actions, arguments, and guards in conditional-preference expressions already evaluated. Consequently, the priority criterion selected for a specific action or the information considered for such selection could change. As this information is an important step in the argumentative reasoning process, it is easy to see that the use of a different priority criterion can change the warrant state of an action's preconditions affecting clearly its applicability. The two possible strategies proposed here for solving the new type of threat considered in this section will be used in an algorithm presented next.

6.3. The P-APOP Algorithm

Next, we present an extension of the APOP algorithm, called P-APOP (*Argumentative Partial Order Planning with Preferences*), that includes conditional-preference expressions formalized before. Operationally speaking, the P-APOP algorithm makes use of several functions.

The primary function P-APOP starts with an initial plan and seeks to complete it with new steps, resolving the threats that could appear. Thus, this algorithm firstly builds a null plan and then attempts to complete it with the recursive procedure COMPLETE_PLAN. To represent a plan, we use six sets containing (see MAKE_NULL_PLAN in Algorithm 2): action steps, argument steps, ordering constraints, subgoals, causal links, and support links.

To achieve its goal, function COMPLETE_PLAN (see Algorithm 3) uses two auxiliary functions during the involved process:

- GET_STEPS (see Algorithm 4): this function will build plan steps to support an unsatisfied subgoal. Note that the process to find an argument to support a subgoal is achieved by using a priority criterion obtained from a selected path structure. If no argument can be constructed, then only actions are considered.

  Following the notation of [7], we use "*possibly* $S_i \prec S_j$" in several of our algorithms to represent that $S_i \prec S_j$ is consistent with the ordering constraints of the corresponding plan ($Plan.OC$). Let us denote $Plan.OC^+$ the transitive closure of $Plan.OC$. Then, we will say that $S_i \prec S_j$ is consistent with $Plan.OC$, if $Plan.OC \cup \{S_i \prec S_j\}$ is a strict partial order, or equivalently $S_j \prec S_i \notin Plan.OC^+$. In addition, we will say that $S_i \prec S_k \prec S_j$ is consistent with $Plan.OC$, if $Plan.OC \cup \{S_i \prec S_k, S_k \prec S_j\}$ is a strict partial order.

- RESOLVE_THREATS (see Algorithm 5): this function attempts to resolve the threats that could appear when a new step is added. In particular, function GUARD_ACTION_THREAT (see Algorithm 6) resolves guard-action threats using methods *promotion* and *disabling literal* proposed in Section 6.2. Checking the existence of threats every time a new step is added in the plan allows us to avoid being trapped in loops of adding subgoals that were already considered in the planning process. Threats that involve only actions and arguments are out of the scope of this paper, as they are essentially the same as in APOP. A description of the algorithms that deal with these problems can be found in [6, 7].

Algorithms 1, 2, and 3 implement an outline of the P-APOP algorithm and proceed in a manner similar to that of the traditional APOP algorithm. Besides the initial state Ψ and the goals $\mathbb{G}$, function P-APOP considers Δ and $\mathbf{A}$ as input parameters. The set Δ contains defeasible rules whose weights possibly change by the use of a different priority criterion when new action steps are added to the plan. For convenience, we will assume 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 selects an unsatisfied subgoal:

$$(SubGoal, Step, SubGoalType, ArgLine, Crit)$$

where, *ArgLine* is an argumentation line associated with *SubGoal*, and one of the following conditions is to be satisfied:

Algorithm 1 P-APOP algorithm

```
function P-APOP($\overset{\downarrow}{\Psi}, \overset{\downarrow}{\Delta}, \overset{\downarrow}{\mathbb{G}}, \overset{\downarrow}{\mathbf{A}}$): $Plan$
    $Plan :=$ MAKE_NULL_PLAN($\Psi, \mathbb{G}$);
    $Plan :=$ COMPLETE_PLAN($Plan, \Delta, \mathbf{A}, \Psi$);
    return $Plan$;
end function
```

Algorithm 2 Function to make null plan

```
function MAKE_NULL_PLAN($\overset{\downarrow}{\Psi}, \overset{\downarrow}{\mathbb{G}}$): $Plan$
    $Plan.ActS := \{(start, \emptyset, \Psi, \emptyset), (finish, \mathbb{G}, \emptyset, \emptyset)\}$;        ▷ Action steps
    $Plan.ArgS := \emptyset$;                                                  ▷ Argument steps
    $Plan.OC := \{start \prec finish\}$;                                       ▷ Ordering constraints
    $Plan.SG := \{(G, finish, \emptyset, ac\_arg, [\ ]) | G \in \mathbb{G}\}$;   ▷ Subgoals or open conditions
    $Plan.CL := \emptyset$;                                                    ▷ Causal links
    $Plan.SL := \emptyset$;                                                    ▷ Support links
    return $Plan$;
end function
```

- there exists an action step $(Step, \_, \mathsf{P}, \Gamma)$, such that $SubGoal \in \mathsf{P}$, $SubGoalType \in \{ac, arg, ac\_arg\}$[5], and $Crit$ is the priority criterion associated with Γ.

- there exists an argument step $(Step, \langle \mathcal{B}, (P; \omega) \rangle, \_, \mathsf{prc})$, such that $SubGoal \in Base(\mathcal{B})$, $SubGoalType = ac$, and $Crit = \mathsf{prc}$.

Then, the set of steps to achieve the subgoal in consideration is obtained via the execution of function GET_STEPS.

Algorithm 4 is in charge of constructing action steps (apart from argument steps) to support an unsatisfied subgoal

$$(P, S_j, SubGoalType, [\langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_n, L_n \rangle], \mathsf{prc}_j).$$

The set *Act_Steps* contains all action steps where P is an action's effect. These steps can be either actions in $\mathbf{A}$, or action steps appearing in the plan before S_j. Moreover, *Arg_Steps* constitutes a set of argument steps containing all potential arguments for P built from Δ_{prc_j}.

[5] *SubGoalType* determines the type of step that must be used to achieve *SubGoal*. The values ac, arg, and ac_arg indicate the need for an action step, an argument step, or either of the two types, respectively. For instance, if a new action step $(\mathsf{N}, \mathsf{X}, \mathsf{P}, \Gamma)$ is added in the plan, then $SubGoal = ac\_arg$ for each $SubGoal \in \mathsf{P}$; this allows to use either an action step or an argument step to achieve each precondition of N. Observe also that if there exists an argument step, then *ArgLine* is updated with the argument at hand (see Algorithm 4).

Algorithm 3 Function to complete plan

```
function COMPLETE_PLAN($\overset{\downarrow\uparrow}{Plan}, \overset{\downarrow}{\Delta}, \overset{\downarrow}{\mathbf{A}}, \overset{\downarrow}{\Psi}$): $Plan$
    if $Plan.SG = \emptyset$ then return $Plan$;
    end if
    choose $(SubGoal, Step, SubGoalType, ArgLine, Crit) \in Plan.SG$ from $Plan.SG$;
    $Plan.SG := Plan.SG \setminus (SubGoal, Step, SubGoalType, ArgLine, Crit)$;
    $Steps :=$ GET_STEPS$(SubGoal, Step, SubGoalType, ArgLine, Crit)$;
    for each $S \in Steps$ do
        if $S = (S_i, Pre_i, Ef_i, (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E_i})$ then
            $Plan.CL := Plan.CL \cup \{S_i \xrightarrow{P} S_p\}$;
            $Plan.OC := Plan.OC \cup \{S_i \prec S_p\}$;
            if $S \notin Plan.ActS$ then
                $Plan.ActS := Plan.ActS \cup (S_i, Pre_i, Ef_i, (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E_i})$;
                $Plan.OC := Plan.OC \cup \{start \prec S_i, S_i \prec finish\}$;
                $Plan.SG := Plan.SG \cup \{(G, S_i, S_p, ac\_arg, [\,], \mathsf{prc}) \mid G \in Pre_i\}$;
            end if
        end if
        if $S = (S_i, \langle \mathcal{B}_i, (P; \omega)\rangle, ArgLine_i, \mathsf{prc}_j)$ then
            $Plan.ArgS := Plan.ArgS \cup (S_i, \langle \mathcal{B}_i, (P; \omega)\rangle, ArgLine_i, \mathsf{prc}_j)$;
            $Plan.OC := Plan.OC \cup \{S_i \prec Step\}$;
            $Plan.SL := Plan.SL \cup \{S_i \overset{P}{\leftharpoondown} Step\}$;
            $Plan.SG = Plan.SG \cup \{(G, S_i, S_p, ac, ArgLine_i, \mathsf{prc}_j) \mid G \in Base(\mathcal{B}_i)\}$;
        end if
        if RESOLVE_THREATS$(Plan, \Delta) =$ null then
            REMOVE $S$ FROM $Plan$;
        else
            $\pi :=$ COMPLETE_PLAN$(Plan, \Delta, \mathbf{A}, \Psi)$;
            if $\pi \neq$ null then
                return $Plan$;
            end if
            if STOPPOINT_BACKTRACKING$(S) =$ false then
                return null;
            end if
        end if
    end for
    REMOVE $Step$ FROM $Plan$;
    UPDATE_STOPPOINT_BACKTRACKING$(Step)$;
    return null;
end function
```

Algorithm 4 Function to get steps

```
1: function GET_STEPS($\overset{\downarrow}{SG}$): $Steps$
2:     Let $SG = (P, S_j, SubGoalType, [\langle \mathcal{A}_1, L_1\rangle, \ldots, \langle \mathcal{A}_n, L_n\rangle], \mathsf{prc}_j)$;
3:     $Act\_Steps := \{(S_{ac}, Pre, Ef, (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi, E_{ac}}) \mid$
         $P \in Ef, (S_{ac}, Pre, Ef, (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi, E_{ac}}) \in Plan.ActS$
         THAT $possibly$ $S_{ac} \prec S_j or S_{ac}$ IS CREATED USING AN ACTION $\mathsf{A} \in \mathbf{A}\}$
           WHERE $(\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi, E_{ac}}$ IS OBTAINED FROM $\Psi\}$;
4:     $Arg\_Steps := \{(S_{arg}, \langle \mathcal{B}, (P;\omega)\rangle, ArgLine_{arg}, \mathsf{prc}_j) \mid \langle \mathcal{B}, (P;\omega)\rangle$
         IS A POTENTIAL ARGUMENT FOR $P$ FROM $\Delta_{\mathsf{prc}_j}$ ACCEPTABLE W.R.T.
           $[\langle \mathcal{A}_0, (L_0;\omega_0)\rangle, \ldots, \langle \mathcal{A}_n, (L_n;\omega_n)\rangle], ArgLine_{arg} =$
           $[\langle \mathcal{A}_0, (L_0;\omega_0)\rangle, \ldots, \langle \mathcal{A}_n, (L_n;\omega_n)\rangle, \langle \mathcal{B}, (P;\omega)\rangle]\}$;
5:     switch $SubGoalType$ do
6:         case $ac$ : $Steps := Act\_Steps$;
7:         case $arg$ : $Steps := Arg\_Steps$;
8:         case $ac\_arg$ : $Steps := Act\_Steps \cup Arg\_Steps$;
9:     return $Steps$;
10: end function
```

Once the set *Steps* has been built, Algorithm 3 checks for the existence of an appropriate step to achieve *SubGoal* according to the value of *SubGoalType*. The statement **for each** in line 7 allows the algorithm to choose among different alternative steps. In each case, the plan will be updated accordingly.

In contrast with APOP, the set of action steps in a plan (*Plan.ActS*) includes in each of its elements a selected path structure. This component is essential in the planning process for two reasons: 1) deciding which priority criterion should be used at the moment a warrant for an action's preconditions is needed (see Algorithm 4), and 2) identifying all guard-action threats (see Algorithm 6). Another difference is that an argument step

$$(S_i, \langle \mathcal{B}_i, (P;\omega)\rangle, ArgLine_i, \mathsf{prc}_j)$$

includes the criterion that should be considered later by a defeater for $\langle \mathcal{B}, (P;\omega)\rangle$ that could appear as the plan is built. Finally, our algorithm includes in each subgoal of *Plan.SG* the priority criterion upon which the reasoning over an action's preconditions will be based.

The incorporation of an argument step to a plan to support an unsatisfied subgoal of an action step is not enough to warrant that precondition since the potential argument associated with the argument step could be defeated by other ones. At this point, it is not possible to know all defeaters of an argument since they depend on effects of action steps added later in the planning process. For this reason the dialectical tree for a particular argument is built gradually as the plan is built. The component *ArgLine* in a subgoal is essentially used as a way to arrive at acceptable argumentation lines associated with a particular potential argument. Recall that the existence of a new defeater

Algorithm 5 Function to resolve threats

```
1: function RESOLVE_THREATS($\overset{\downarrow\uparrow}{Plan}, \overset{\downarrow}{\Delta}, \overset{\downarrow}{\Psi}$): $Plan$
2:     if ACTION_ACTION_THREAT($Plan$) $\neq$ null then
3:         if ACTION_ARGUMENT_THREAT($Plan$) $\neq$ null then
4:             if ARGUMENT_ARGUMENT_THREAT($Plan, \Delta$) $\neq$ null then
5:                 if GUARD_ACTION_THREAT($Plan, \Psi$) $\neq$ null then
6:                     return $Plan$;                         ▷ No threats exist
7:                 end if
8:             end if
9:         end if
10:    end if
11:    return null;                          ▷ There exists an unresolved threat
12: end function
```

gives rise to a new threat *argument-argument* that will be then addressed by function ARGUMENT_ARGUMENT_THREAT (see Algorithm 5). Besides identifying defeaters interfering with a potential argument, this function will resolve such threats attempting particularly to warrant those arguments that support a precondition of an action step. Keeping *ArgLine* updated turns out to be an essential task to address this issue. We refer the reader to the APOP algorithm [7] for further details about how *ArgLine* is updated when a defeater is identified.

As we have already mentioned, after a new step is added to the plan, new threats could occur. The procedure GUARD_ACTION_THREAT will consider all action steps in the plan to detect possible interferences with guards of conditional expressions already evaluated. Note that statement $S_G = S_{GS} \cup S_{GNS}$ in Algorithm 6 identifies all these possible interference cases for every $(\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E}$ extracted from each evaluated expression E. Then, the procedure tries to resolve each of them, choosing either *promotion* or *disabling literal*. The procedure will return `null` in case it encounters an unresolvable threat. Apart from guard-action threats, Algorithm 5 considers—with the first three statements—the threats usually present in APOP that involve only actions and arguments (its implementation could be made in an analogous way to [7]). Although specifics of such implementations are outside the scope of this work, we assume that they will return `null` when they cannot resolve the threat in question; otherwise, they will return the current plan.

Once RESOLVE_THREATS has been executed, Algorithm 3 checks the existence of unresolved threats. Note that unresolved threats involve backtracking, which implies removing step S (along with its dependencies) from *Plan*, and considering pending alternatives. Finally, 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 choice was made. If this choice is a backtrack point, the pending alternatives are considered. 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,

Algorithm 6 Function to resolve guard-action threats

```
function GUARD_ACTION_THREAT($\overset{\downarrow\uparrow}{Plan}$, $\overset{\downarrow}{\Psi}$): $Plan$
    for each $(S_j, \_, \_, (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E}) \in Plan.ActS$ do
        $S_{AL} := \{(A, L) \mid A \in Plan.ActS \wedge L \in Ef_A \wedge$ possibly $A \prec S_j \wedge \nexists D \in$
        $Plan.ActS \wedge \overline{L} \in Ef_D \wedge A \prec D \prec S_j\}$;
        $S_L := \{L \mid (A, L) \in S_{AL}\}$;
        $S_{GS} = \{(A, L) \mid (A, L) \in S_{AL} \wedge \overline{L}$ IS PRESENT IN $\mathcal{G}^+\}$
        for each $\gamma \in \mathcal{G}^-$ do
            $L_{NS} := \{L \mid L \in \gamma \wedge L \notin \Psi\}$;
            if $L_{NS} \subseteq S_L$ then
                choose $N$ from $L_{NS}$;
                $S_{GNS} := S_{GNS} \cup \{(A, N) \mid (A, N) \in S_{AL}\}$;
            end if
        end for
        $S_G = S_{GS} \cup S_{GNS}$
        for each $(S_i, N) \in S_G$ do
            choose either
                if possibly $S_j \prec S_i$ then                                ▷ Promotion
                    $Plan.OC = Plan.OC \cup \{S_j \prec S_i\}$;
                else return null;
                end if
                if $S_i \xrightarrow{N} S_k \in Plan.CL \wedge S_k \prec S_j$ then                ▷ Disabling
                    Let $S_k = (S_k, Pre, Ef, (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E_k})$;
                    $Plan.SG = Plan.SG \cup (\overline{N}, S_k, ac, [\,], (\mathcal{G}^+, \mathcal{G}^-, \mathsf{prc})_{\Psi,E_k})$;
                else return null;
                end if
            end choose
        end for
    end for
end function
```

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. In our algorithm, we consider two auxiliary functions to help the backtracking process. In particular, UPDATE_STOPPOINT_BACKTRACKING updates the point to backtrack to in order to try a different choice, every time a step fails, and STOPPOINT_BACKTRACKING is used to check whether the current step is a point to make the backtracking stop. This is illustrated in Figure 5, which shows two backtracking steps. The first occurs because no step can be constructed to support some of the literals that are present in the base of argument $\langle \mathcal{A}, (e; \omega) \rangle$. Backtracking stops at finding the point where the choice of $\langle \mathcal{A}, (e; \omega) \rangle$ was made. Then, another backtrack is triggered because the precondition e fails to be satisfied. Note that every time a backtrack to make a new choice begins, the current step and its dependencies are removed from the plan, as shown in Figure 5-(b) and 5-(c) with steps $\langle \mathcal{A}, (e; \omega) \rangle$ and N_i, respectively.

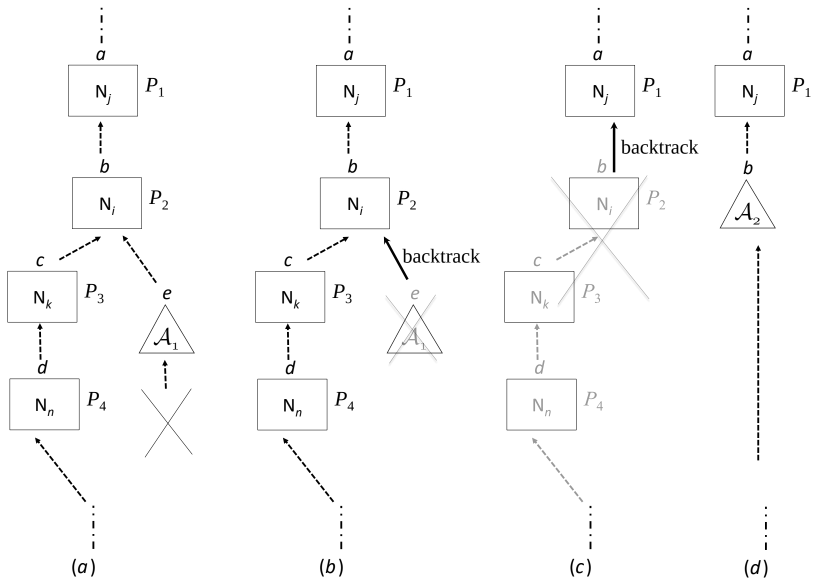


Figure 5: Backtracking example.

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 P-APOP is sound and complete[6], essentially because a suc-

[6]Soundness here is defined as the property that ensures that applicable sequences of actions obtained from a partial plan generated by the algorithm are such that, when executed in an

cessful search terminates with a partially-ordered solution plan, *i.e.,* any linear plan that satisfies the partially ordered one is such that all action step preconditions are necessarily satisfied, and backtracking ensures that the search space is eventually exhausted.

7. Complexity Results

We have seen how defeasible argumentation frameworks like P-DeLP can be combined with partial-order planning techniques to consider arguments as planning steps, and in Section 6.3 we presented the P-APOP algorithm for constructing plans. In this section, we address a major issue that arises in plan construction processes, which is that they must satisfy reasonable response time requirements.

Therefore, we now focus on analyzing the complexity of our approach in order to determine in which cases it is suitable for supporting the development of real-world applications; throughout this analysis, we first focus on identifying factors that affect the performance of the main P-APOP algorithm (Algorithm 1), and then on the data and combined complexity of query answering in the context of P-DeLP. Before presenting our results, we briefly recall some basic concepts; a thorough introduction to these notions is outside the scope of this paper, and we refer the interested reader to [30, 31] for further details.

- *Computational cost* is most commonly analyzed in terms of the time and/or space needed to arrive at a solution to a particular problem. In order to do so, the *instance size* needs to be characterized in terms of parameters of interest; these parameters should be adequately chosen to represent how problem instances grow.

- *Running time* (also known as *algorithmic complexity*) refers to a specific algorithmic solution to a problem, and is typically analyzed using *asymptotic notation* (proposed over a century ago by Bachmann and Landau [32, 33]) to identify a family of functions that characterizes how computational cost grows as instance size grows. Examples of common running time expressed in this manner is "$O(n^2)$" or "$O(n \log n)$"—these are written in *Big-O* notation, a particular kind of asymptotic notation.

- The term *computational complexity* (typically abbreviated simply as *complexity*) usually refers to the difficulty of solving a *problem* (i.e., without necessarily considering specific algorithms to do so). To abstract away unnecessary details, complexity is often studied for *decision* problems, which are those that have a Yes/No answer. In our case, an example of a decision problem is the main one we will be studying: *Does there exist a plan P such that, executed starting in state Ψ, arrives at goal $\mathbb{G}$ following*

initial state, will lead to satisfying all specified goals. Completeness in our scenario is defined as the algorithm's capability of finding all possible solutions.

priorities **C** *and satisfies the constraints imposed by* Δ *and* **A***?* Examples of decision problem complexities are "NP-hard" or "PTIME", where the former is a *hardness* result (sometimes referred to as a lower bound), and the latter a membership one (sometimes referred to as an upper bound). When a problem is both a member of a class C and hard for the same class, it is said to be "C-complete".

Data and Combined Complexity. A common refinement of complexity analysis was proposed by Vardi (in the context of databases, but it is now commonly used in many areas) [34] and assumes restricted classes of instances in which certain parts of the problem are fixed. Here, we will make use of the notions of *data complexity*—which refers to the complexity of query answering when only Ψ is considered to be an input (that is, Δ, $\mathbb{G}$, and **A** are considered fixed)—and *combined complexity* when all parameters are allowed to vary. Other commonly used refinements, such as query, fixed program, and bounded arity complexity will be studied in future work.

Next, we begin by examining several factors in light of the main procedures that have a major impact on the performance of Algorithm 1.

7.1. Main factors affecting the running time of our algorithm

In our proposal, we have particularly focused on partial order planning (POP); much work has been done on comparing the computational cost of different POP algorithms, but little attention has been devoted to analyzing time complexity in defeasible planning. Our aim in this section is to address this issue by focusing on analyzing several main factors that affect the performance of our algorithm. Although determining the precise asymptotic running time of the main P-APOP algorithm is outside of the scope of this work, this analysis provides sufficient details for a preliminary study of the computational cost of our approach; in particular, it is helpful in identifying the *complexity sources*, which guide the study in the next section.

As a starting point in the endeavor to better understand the computational cost of deriving a plan in our proposal, we now discuss a few main differences in comparison to classical POP algorithms, focusing on the subroutines that have the largest effect on the overall running time. To facilitate this study, we will briefly recall some of the algorithms previously introduced in this paper, such as COMPLETE_PLAN (Algorithm 3), and its auxiliary functions GET_STEPS (Algorithm 4) and RESOLVE_THREATS (Algorithm 5). Next, we outline the running time of the following procedures, which are the basis of obtaining a single *partial plan*:

— *Building an argument:* The procedure GET_STEPS is in charge of constructing argument steps to support a precondition. Building an argument depends on checking the existence of a potential argument for such precondition from a set of given defeasible rules. Verifying every subset of Δ and checking all conditions of a potential argument definition can be accomplished in polynomial time (for fixed Δ) in the size of the input.

— *Obtaining a selected path structure:* This structure is generated after a conditional expression is evaluated, that is by applying a recursive procedure to support the selection of a priority criterion. Note that conditional preference expressions can be represented via a full binary tree where every inner node is labeled with a guard and each leaf node is labeled with a criterion. The recursive loop only requires $O(\mathsf{n})$ time, where n is the maximum number of guards in the search tree from the root guard to the selected criterion.

— *Resolving an action-argument threat:* Identifying and resolving a threat depends on the number of causal links in the plan, which we denote with cl. Assuming the worst case, each link is analyzed as many times as action steps have been generated during the planning process; thus, this also depends on the number as of action steps present in the plan. Since the cost of resolving the threat is linear, the total time demanded will be in $O(\mathsf{cl} * \mathsf{as})$.

— *Resolving an argument-argument threat:* When creating an argument step for a precondition, defeaters interfering with such argument could appear as a new threat. The time for checking the existence of a defeater is dominated by the number of defeasible rules; this step can be carried out in polynomial time when Δ is fixed.

— *Resolving a guard-action threat:* There exists a selected path structure associated with each action step in the plan, and identifying this type of threat involves visiting all steps. Then, it is necessary to look at those action steps in the plan capable of changing the satisfaction or insatisfaction state of a guard, and this can be done in polynomial time.

Towards a more concrete result, analyzing the performance of a planning algorithm requires looking both at the *number of plans* generated when solving a planning problem and the computational cost of *obtaining each plan.* In our discussion so far, we showed that the cost per plan in our approach is strongly influenced by several factors that affect overall performance, but in order to obtain a complete analysis we must consider the number of plans generated until a solution plan is found. Several research efforts have been conducted following this direction; for example, Knoblock and Yang [35] proposed to study the planning process as a search through a space of plans—their approach represents the search space as a tree, where each inner node corresponds to a plan and each arc corresponds to a plan transformation. In our proposal, each transformation would extend a plan by adding additional steps in addition to ordering constraints or new subgoals when a new threat is addressed. In turn, each leaf in the search tree corresponds either to a solution plan or a dead-end, and each intermediate node corresponds to a plan that can be extended.

The intuitions in [35] can also be applied in our problem, in this case as a particular type of depth-first search through a space of plans. This leads to challenging problems such as determining the branching factor and depth of search, which are outside the scope of this initial study. However, in the next section, we can already see that the outlook is not promising since these in general have a crippling effect, which is reflected in the result that our planning problem is PSPACE-hard and in EXPTIME in the combined complexity. On a more positive note, we also show that for several fragments—where restrictions are appplied—the complexity is lower.

7.2. Main complexity results

As we mentioned previously, the problem of studying the complexity of argumentation-based epistemic planning is challenging, and has not yet been fully addressed. In this section, we provide several complexity results for the following central problem:

> P-APOP Solution Existence: 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 $\mathbf{A}$?

We sometimes refer to P-APOP Solution Existence as *the preference-based planning problem* or *our planning problem.* In order to explore the different sources of complexity, we will study a variety of restricted versions or fragments of the problem as well—these results can be summarized as follows (cf. Figure 6):
P-APOP Solution Existence is:

— PSPACE-hard and in EXPTIME in the general case (no limits imposed on action effects and preconditions).

— DP-hard in the combined complexity if: (i) actions are allowed to have a single precondition, or (ii) actions are allowed to have a single positive precondition and up to two postconditions.

— NP-complete in the data complexity if either: (i) actions are restricted to positive postconditions, or (ii) actions can have a single precondition and a single positive postcondition.

— In PTIME in the data complexity if actions are restricted to either: (i) positive preconditions and one postcondition, (ii) a single precondition and the number of goals is bounded by a constant, or (iii) no preconditions.

In [36], the authors provide several complexity results for PLANSAT—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 main complexity contributions for DeLP reported in [37], will be used as the bases for our analysis in the rest of the section.

Table 6 illustrates the main complexity results of our work; it includes the hierarchy of different planning problems presented in [36], and shows how the computational complexity varies from PTIME to EXPTIME, depending on different restrictions that can be considered. The results for PLANSAT are summarized in the second column, whereas the main data and combined complexity

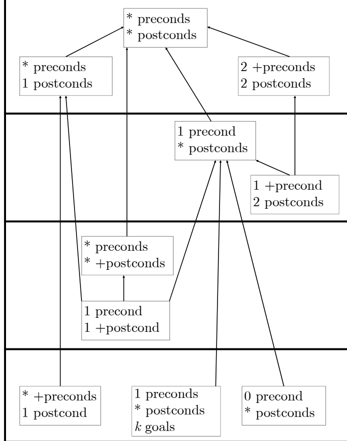
Hierarchy of Planning Problems	Complexity		
	Planning	**P-DeLP**	**Preference-based planning**
 * preconds * postconds * preconds 1 postconds 2 +preconds 2 postconds 1 precond * postconds 1 +precond 2 postconds * preconds * +postconds 1 precond 1 +postcond * +preconds 1 postcond 1 preconds * postconds k goals 0 precond * postconds	PSPACE-complete	EXPTIME [combined]	EXPTIME, PSPACE-hard [combined] (Prop. 1)
	NP-hard	co-NP-hard [combined]	DP-hard [combined] (Prop. 2)
	NP-complete	NP [data]	NP-complete [data] (Prop. 3)
	PTIME	PTIME [data]	PTIME [data] (Prop. 4)

Figure 6: Overview of complexity results: The figure in the left hand side is reproduced from [36], as are the results in the "Planning column"; the results in the "P-DeLP" are direct consequences from those in [37].

results for P-DeLP (the decision problem of whether a literal is warranted) are given in the third column. Finally, the last column gives the complexity results for our planning problem under each set of restricted versions.

We define data and combined complexity in the context of P-DeLP following the approach taken for DeLP in [37]. Making an analogy with the corresponding concepts in databases, the current state Ψ represents the input database; the data complexity of the decision problem to check whether a literal is warranted is therefore the complexity of the problem when the set of defeasible rules Δ is fixed, while the combined complexity corresponds to the case in which both Ψ and Δ vary. In our case, it is immediate to check that the data complexity of this problem for P-DeLP is the same as that in the DeLP case [37], *i.e.,* in some class NP^C, where C is the class determined by the particular argument preference criterion. From the developments in [37], we have membership in EXPTIME for combined complexity.

Proposition 1. P-APOP SOLUTION EXISTENCE *is PSPACE-hard and in EXPTIME in the combined complexity.*

Proof In [36], the PLANSAT problem is shown to be PSPACE-complete. This propositional planning problem is a particular type of preference-based planning problem, given that the latter adds defeasible rules—and preference criteria over them—to the classical domain. Therefore, the PSPACE-hardness result in [36] can almost directly de transferred to our case since we only need to observe that in instances in which the set Δ (and consequently $\mathbf{C}$) is empty, we arrive at PLANSAT instances, so PLANSAT can trivially be reduced to our problem. This shows that our preference-based planning problem is PSPACE-hard in the combined complexity, which establishes a lower bound on the problem's complexity. To establish an upper bound, we must look at the added cost of carrying out the computations associated with P-DeLP; since it is known that the worst-case computational cost of deciding warrants in DeLP is exponential [37], and that P-DeLP additionally involves updating weights—which can clearly be done with an EXPTIME budget—it follows that out problem is in EXPTIME (in the combined complexity). □

In [36], Bylander examines how computational complexity for planning varies depending on a variety of conditions, and some interesting results are obtained. We now follow the same path for studying fragments for the P-APOP case and derive several interesting results.

Proposition 2. P-APOP SOLUTION EXISTENCE *is DP-hard in the combined complexity whenever either: (i) actions are allowed to have a single precondition, or (ii) actions are allowed to have a single positive precondition and up to 2 postconditions.*

Proof The class DP is defined as the set of all languages $L = L_1 \cap L_2$, where $L_1 \in$ NP and $L_2 \in$ co-NP [38]. In [36], the PLANSAT problem is shown to be NP-hard whenever either condition (i) or (ii) hold (referred to as PLANSAT+), while deciding whether a literal is warranted given a DeLP program is known to be co-NP-hard [39].

In our setting, the preference-based planning problem follows this arrangement, since it must satisfy both the conditions imposed by PLANSAT+ (finding a sequence of actions that satisfy the goals) and DeLP warrants (action preconditions). Therefore, we can reduce SAT-UNSAT, a DP-complete problem [38], to preference-based planning making use of the same reductions that lead to the hardness results mentioned above, with a simple addition as follows. We begin with two Boolean formulas F_1 and F_2, and must create an instance of PLANSAT and a P-DeLP program such that the planning problem encodes F_1 and the P-DeLP program encodes F_2. We do this as follows:

- Use the reduction from SAT to PLANSAT+ in [36] to obtain an P-APOP problem instance (Δ is empty).
- Add a fresh action α (i.e., an action that does not appear in the problem), with a single postcondition ϕ that is also added as precondition to all other actions in the instance. Therefore, all solutions must begin with action α; therefore, it is easy to see that this does not interfere with the reduction from SAT to PLANSAT+.
- Use the reduction from UNSAT to the problem of determining the warrant status of a literal in DeLP in [39]. This reduction introduces an arbitrary

literal whose warrant status is tied to the (un)satisfiability of the UNSAT instance; to achieve our composite reduction, we can simply use the same literal ϕ introduced as the postcondition of α in the previous step.

From this construction, it is clear that F_1 is satisfiable and F_2 is not if and only if the preference-based planning instance derived from the two reductions has a solution, which concludes the DP-hardness proof. □

The following proposition states that our problem is NP-complete in the data complexity under certain conditions.

Proposition 3. P-APOP Solution Existence *is NP-complete in the data complexity when actions are restricted to any number of effects but without negated literals.*

Proof The computational complexity of PLANSAT limited to actions with any number of effects without negated literals is NP-complete. Any PLANSAT instance can be polynomially reduced to a preference-based planning problem instance if the set of defeasible rules Δ is empty, which implies that P-APOP Solution Existence is NP-hard in the data complexity. Since deciding the warrant status of a literal in P-DeLP is NP in the data complexity, we can conclude that our problem is also in NP. Therefore, our planning problem is NP-complete in the data complexity. □

According to the results shown in [36], PLANSAT is polynomial under several different sets of assumptions; next, we show that such restrictions can be incorporated into our problem so that it can also be solved in polynomial time if the set Δ is fixed. This leads to the following proposition.

Proposition 4. P-APOP Solution Existence *is in PTIME in the data complexity under any of the following restrictions:*

— *Number of goals at most k, and actions restricted to one precondition.*

— *Preconditions have no negated literals, and only one effect.*

— *Actions restricted to no preconditions.*

Proof For fixed Δ, the decision problem of checking whether a literal is warranted is in PTIME in the data complexity [37]. Given that the number of arguments is polynomial when Δ is fixed, and determining whether a literal is warranted can be done in polynomial time when the argument preference criteria is polynomial, our problem is thus also solvable in polynomial time under these restrictions, which proves membership in PTIME in the data complexity. □

The results obtained in this section, in general terms, emphasize the fact that studying both the data and combined complexities for P-DeLP allowed us to analyze the difficulty of solving our planning problem under a variety of conditions. As a final remark, note that we have not specifically considered the complexity of computing the argument preference criterion; though there exist several cases that can be studied, we point out that computing generalized specificity—the default criterion used in (P-)DeLP—can clearly be done in polynomial time; see [15] for further details on this criterion.

8. Application Example

In this section, we will apply our approach to a scenario where a cooking service robot must prepare a meal considering the homeowners' particular preferences. The robot can perform several tasks, such as setting the table, cooking, or calling a food delivery service, and its job finishes when the meal is ready on the table. Our approach expands the robot's planning system capacity by considering contextual preferences, *i.e.,* user's preferences expressed in a particular context. Note that we will build on this simplistic scenario essentially to illustrate and apply our proposal—a formal definition of the robot's characteristics is out of the scope of this example.

Consider the following set of facts representing the world state:

$$\Psi_8 = \left\{ \begin{array}{l} \mathit{suggested\_on\_TV}(f1) \\ \mathit{uncommonly\_consumed}(f1) \\ \mathit{friend}(\mathit{maria}) \\ \mathit{like}(\mathit{maria}, f1) \\ \mathit{near\_dService}(\mathit{deliv}1) \\ \mathit{good\_food}(\mathit{deliv}1) \\ \mathit{bad\_service}(\mathit{deliv}1) \\ \mathit{cooking\_recipe}(f1) \\ \mathit{delivery\_service}(\mathit{deliv}1) \\ {\sim}\mathit{homemade\_food} \\ {\sim}\mathit{food\_ordering} \end{array} \right\}$$

The set Ψ_8 contains information related to foods and delivery services that a system can consider during the planning process. Literals $\mathit{suggested\_on\_TV}(f1)$ and $\mathit{uncommonly\_consumed}(f1)$ express that $f1$ is a food suggested by several TV programs, and it is not a commonly consumed food, respectively. Also, the literals $\mathit{friend}(\mathit{maria})$ and $\mathit{like}(\mathit{maria}, f1)$ state that *maria* is a good family friend, and she has expressed that $f1$ is one of her most preferred foods. Moreover, the literals $\mathit{near\_dService}(\mathit{deliv}1)$ and $\mathit{good\_food}(\mathit{deliv}1)$ indicate that the delivery service $\mathit{deliv}1$ is located nearby and offers good food, but its customer service is bad ($\mathit{bad\_service}(\mathit{deliv}1)$). Moreover, the literals $\mathit{cooking\_recipe}(f1)$ and $\mathit{delivery\_service}(\mathit{deliv}1)$ convey that there exists an available recipe for making $f1$ and $\mathit{deliv}1$ is a food delivery service. Finally, ${\sim}\mathit{homemade\_food}$ and ${\sim}\mathit{food\_ordering}$ express that there is no homemade food or food orders.

In order to model the robot's knowledge, we consider the following set of defeasible rules:

$$\Delta_8 = \left\{ \begin{array}{l} (\mathit{preferF}(F) \leftarrow \mathit{friend}(FF), \mathit{like}(FF, F);\ 0.8) \\ ({\sim}\mathit{preferF}(F) \leftarrow \mathit{friend}(F), \mathit{dislike}(FF, F);\ 0.7) \\ (\mathit{preferF}(F) \leftarrow \mathit{suggested\_on\_TV}(F);\ 0.5) \\ ({\sim}\mathit{preferF}(F) \leftarrow \mathit{suggested\_on\_TV}(F), \mathit{infrequently\_consumed}(F);\ 0.3) \\ (\mathit{preferD}(D) \leftarrow \mathit{near\_dService}(D);\ 0.9) \\ ({\sim}\mathit{preferD}(D) \leftarrow \mathit{good\_food}(D), \mathit{bad\_service}(D);\ 0.3) \\ (\mathit{preferD}(D) \leftarrow \mathit{good\_food}(D);\ 0.3) \\ (\mathit{food\_ready} \leftarrow \mathit{homemade\_food};\ 0.1) \\ (\mathit{food\_ready} \leftarrow \mathit{delivery\_service\_food};\ 0.5) \end{array} \right\}$$

The set Δ_8 contains several defeasible rules, the first two of which represent tentative reasons to establish whether food at home is ready to eat: "if there is homemade food" or "if there is delivery service food". The last three rules express reasons for and against establishing which food delivery service is more appealing or preferable: "if D is a nearby delivery service" or "if D offers good food" are both reasons to prefer D, whereas "if D's food is good, but offers poor customer service" is a reason for not preferring D. Observe that the third and fifth rules represent reasons for establishing whether a food is a preferred option: "if F is a food suggested on TV" or "if FF is a family friend and F is considered as one of FF's favorite foods" are both reasons for preferring F. Finally, the fourth and sixth rules can be read as follows: "if F is a food suggested on TV, but infrequently consumed" or "if F is not one of FF's favorite foods" are defeasible reasons against preferring F.

In our scenario, a robot could perform several actions based on the owners' preferences. To analyze each particular action's applicability, here we propose a way of context-adaptable selection of preferences by the use of conditional expressions, as introduced in the previous sections. We present two conditional-preference expressions that implement the following intuitions:

"If it is lunchtime, use Maria's preferences,
otherwise use Juan's preferences", and

"Maria's preferences should be applied".

The intuitions can be captured with the conditional-preference expressions included below:

$E_1 = [\{\mathit{lunchtime}\} : \mathsf{pref\_maria}; \mathsf{pref\_juan}]$, and

$E_2 = \mathsf{pref\_maria}$, where

$$\Delta_{\mathsf{pref\_maria}} = \left\{ \begin{array}{l} (\mathit{preferF}(F) \leftarrow \mathit{friend}(FF), \mathit{like}(FF, F);\ 0.8) \\ ({\sim}\mathit{preferF}(F) \leftarrow \mathit{friend}(FF), \mathit{dislike}(FF, F);\ 0.7) \\ (\mathit{preferF}(F) \leftarrow \mathit{suggested\_on\_TV}(F);\ 0.5) \\ ({\sim}\mathit{preferF}(F) \leftarrow \mathit{suggested\_on\_TV}(F), \mathit{infrequently\_consumed}(F);\ 0.3) \\ (\mathit{preferD}(D) \leftarrow \mathit{near\_dService}(D);\ 0.9) \\ ({\sim}\mathit{preferD}(D) \leftarrow \mathit{good\_food}(D), \mathit{bad\_service}(D);\ 0.3) \\ (\mathit{preferD}(D) \leftarrow \mathit{good\_food}(D);\ 0.3) \\ [\ldots] \end{array} \right\}$$

$$\Delta_{\mathsf{pref\_juan}} = \left\{ \begin{array}{l} (\mathit{preferF}(F) \leftarrow \mathit{friend}(FF), \mathit{like}(FF, F);\ 0.6) \\ ({\sim}\mathit{preferF}(F) \leftarrow \mathit{friend}(FF), \mathit{dislike}(FF, F);\ 0.5) \\ (\mathit{preferF}(F) \leftarrow \mathit{suggested\_on\_TV}(F);\ 0.9) \\ ({\sim}\mathit{preferF}(F) \leftarrow \mathit{suggested\_on\_TV}(F), \mathit{infrequently\_consumed}(F);\ 0.9) \\ (\mathit{preferD}(D) \leftarrow \mathit{near\_dService}(D);\ 0.5) \\ ({\sim}\mathit{preferD}(D) \leftarrow \mathit{good\_food}(D), \mathit{bad\_service}(D);\ 0.9) \\ (\mathit{preferD}(D) \leftarrow \mathit{good\_food}(D);\ 0.8) \\ [\ldots] \end{array} \right\}$$

In our application example, the robot's task consists of preparing a meal and setting the table; also, it can order food from a delivery service. Then, the actions it can perform are the following:

$\mathsf{A}_7 =$

$$\left\{ \begin{array}{l} \{\mathit{table\_ready}\} \xleftarrow{(\mathsf{set\_table}, E_1)} \{\mathit{food\_ready}\} \\ \{\mathit{delivery\_service\_food}\} \xleftarrow{(\mathsf{receive\_food\_delivery}, E_2)} \{\mathit{food\_ordering}\} \\ \{\mathit{food\_ordering}\} \xleftarrow{(\mathsf{order\_food}, E_1)} \{{\sim}\mathit{homemade\_food}, \mathit{delivery\_service}(D), \mathit{preferD}(D)\} \\ \{\mathit{homemade\_food}\} \xleftarrow{(\mathsf{cooking}, E_1)} \{{\sim}\mathit{food\_ordering}, \mathit{food\_recipe}(F), \mathit{preferF}(F)\} \end{array} \right\}$$

They can be interpreted as follows:

— set_table: setting the table. There must be a meal ready to serve.

— receive_food_delivery: receiving food delivery. There must exist a food order.

— order_food: ordering food from a food delivery service. There must not be homemade food kept in the fridge, and the delivery service selected must be a preferable one according to the owner's preferences.

— cooking: cooking at home. There must exist a food recipe available, and the selected food must be chosen according to the owner's preferences.

Given these actions, a preference-based planning problem can be defined as

$$T = (\Psi_8, \Delta_8, \mathbf{C}_8, \mathsf{A}_8, \mathbb{G}_8),$$

where $\mathbb{G}_8 = \{table\_ready\}$ and $\mathbf{C}_8 = \{\mathsf{pref\_maria}, \mathsf{pref\_juan}\}$. In what follows, the sequence of actions $\mathsf{S}_1 = [\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3]$ is analyzed in order to establish whether it is a plan that satisfies the goal *table_ready*.

- Action $\mathsf{A}_1 = \langle \mathsf{X}_1, \mathsf{P}_1, E_1 \rangle$ where

  $\mathsf{A}_1 = \mathsf{order\_food}$

  $\mathsf{X}_1 = food\_ordering$

  $$\mathsf{P}_1 = \{\sim homemade\_food,\ delivery\_service(deliv1),\ preferD(deliv1)\}$$

  $E_1 = [\{lunchtime\} : \mathsf{pref\_maria}; \mathsf{pref\_juan}]$

- Action $\mathsf{A}_2 = \langle \mathsf{X}_2, \mathsf{P}_2, E_2 \rangle$ where

  $\mathsf{A}_2 = \mathsf{receive\_food\_delivery}$

  $\mathsf{X}_2 = delivery\_service\_food$

  $\mathsf{P}_2 = food\_ordering$

  $E_2 = \mathsf{pref\_maria}$

- Action $\mathsf{A}_3 = \langle \mathsf{X}_3, \mathsf{P}_3, E_1 \rangle$ where

  $\mathsf{A}_3 = \mathsf{set\_table}$

  $\mathsf{X}_3 = table\_ready$

  $\mathsf{P}_3 = food\_ready$

  $E_1 = [\{lunchtime\} : \mathsf{pref\_maria}; \mathsf{pref\_juan}]$

For solving the planning problem T, the plan S_1 must be an applicable sequence of actions: since the precondition *preferD(deliv1)* of A_1 is not warranted from $\mathsf{warrL}(\Psi_8, \Delta_{\mathsf{pref\_juan}})$, where $\mathsf{pref\_juan}$ is the priority criterion used after evaluating E_1, S_1 cannot be considered as a solution plan option. Even though it is possible to construct two arguments from Δ_8, $\langle \mathcal{A}_1, (preferD(deliv1); 0.5)\rangle$ and $\langle \mathcal{A}_2, (preferD(deliv1); 0.8)\rangle$ with

$$\mathcal{A}_1 = \{\ (preferD(deliv1) \leftarrow near\_dService(deliv1);\ 0.5)\ \}$$

$$\mathcal{A}_2 = \{\ (preferD(deliv1) \leftarrow good\_food(deliv1);\ 0.8)\ \}$$

supporting $preferD(deliv1)$, are both defeated by $\langle \mathcal{A}_3, (\sim preferD(deliv1); 0.9)\rangle$, where $\mathcal{A}_3 = \{(\sim preferD(deliv1) \leftarrow good\_food(deliv1), bad\_service(deliv1);\ 0.9)\}$. Note that if the conditional preference expression attached to A_1 changes, then S_1 could possibly be a solution plan. Next, we show how the applicability of an action can change depending on the conditional-preference expression used.

Now, we consider a new plan $\mathsf{S}_2 = [\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3]$, but $E_2 = \mathsf{pref\_maria}$ is the preference expression associated with action A_1. Note that it is possible from Δ'_8 to construct the undefeated argument $\langle \mathcal{A}_1, (\mathit{preferD}(\mathit{deliv}1); 0.9)\rangle$ where $\mathcal{A}_1 = \{(\mathit{preferD}(\mathit{deliv}1) \leftarrow \mathit{near\_dService}(\mathit{deliv}1);\ 0.9)\}$, and hence A_1 is applicable in this case. The next two actions, A_2 and A_3, are also applicable. Observe that the only difference between S_1 and S_2 is the priority criterion used for the action A_1; this fact clearly shows the importance of having tools that allow adapting the planning process to the user's preferences when finding a plan to solve the proposed goals.

When the space of plans to achieve the goals is dense, it may be simple to find a solution plan; however, the challenge is to find a solution plan that satisfies the user's needs and preferences. In [40], the authors argue that preferences are of vital importance when the planning problem has too many solutions. The use of preferences contributes to the user's confidence in obtained plans. Given this consideration, we have presented a planning approach for integrating preferences with argumentation-based planners, improving aspects of these planners, such as flexibility for changing the preferences over defeasible knowledge depending on the state of the world and reliability in the generated plans. Finally, Fig. 7 shows how plan S_2 is generated by algorithm P-APOP introduced in Section 6.3.

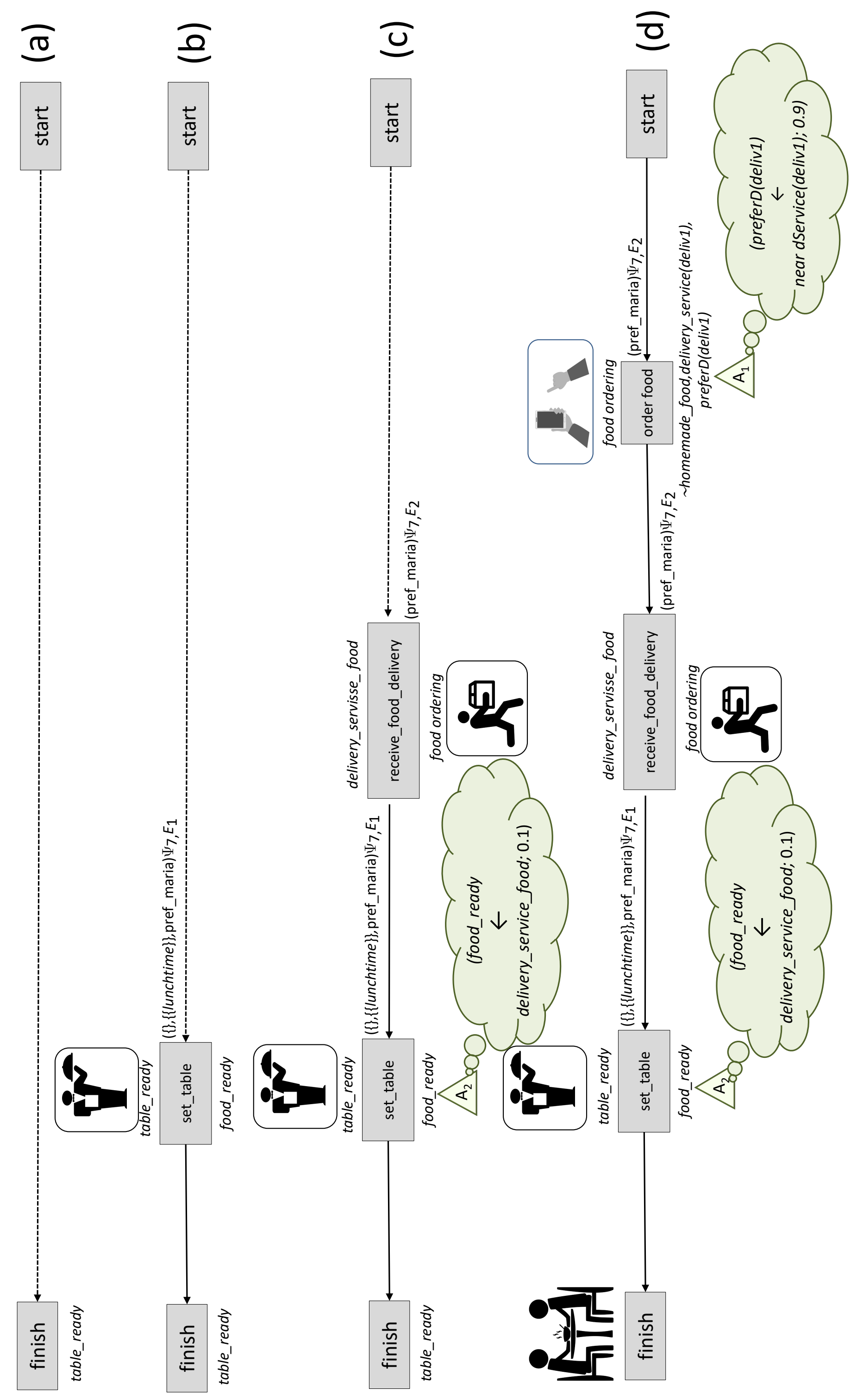

(a)
(b)
(c)
(d)
start
finish
table_ready
set_table
food_ready
({},{{lunchtime}},pref_maria)Ψ_7,E_1
receive_food_delivery
delivery_servisse_food
food ordering
(pref_maria)Ψ_7,E_2
order food
~homemade_food,delivery_service(deliv1), preferD(deliv1)
A_1
A_2
(food_ready ← delivery_service_food; 0.1)
(preferD(deliv1) ← near dService(deliv1); 0.9)

Figure 7: Plan generated using the P-APOP algorithm.

9. Related Work

Combining argumentation and epistemic planning is not a new issue in the literature [41, 7, 42, 8, 43]. However, the approaches in these works have not involved the formalization and use of mechanisms for handling preferences during a planning process. Our approach introduces a programmable mechanism for applying preference in every step (action) of a plan.

In [41], an argumentation formalism for agents following the BDI approach is proposed. The authors present an argumentation-based model to reason about desires (generating desires and plans to achieve them). Similarly to our work, [41] is based on structured argumentation; however, although that approach combines argumentation and plans, there are several differences with our proposal. Firstly, it defines different instantiations of Dungs' abstract argumentation framework for generating consistent desires and consistent plans to achieve these desires. Moreover, it defines the notion of conflict among plans and uses an argumentative analysis to establish which plan prevails. Thus, the notion of a plan is clearly more related to our definition of an argument than to our formalization of a plan. Finally, they use a fixed preference criterion that integrates the value of desires and the cost of plans; instead, we present an approach where several criteria that correspond to priority criteria can be considered. The proposed approach provides mechanisms that allow the selection and change of the contextual priorities that should be used in each plan action.

In [6, 7], an argumentation-based formalism for constructing plans using partial order planning techniques called *DeLP-based partial order planning* (DeLP-POP) is introduced. In this approach, action preconditions can be satisfied by actions' effects or conclusions supported by arguments. The actions and arguments are combined to construct plans. The authors present an extension of the POP algorithm to consider actions and arguments as planning steps and resolve the interferences that can appear. In [8, 44], DeLP-POP is extended to multi-agent cooperative planning; in contrast to our work, they do not focus on how to introduce mechanisms for representing preferences. Our approach extends the representation capabilities of the DeLP-POP system providing the possibility of indicating what preferences, indicated as priority degrees, may be considered at the moment of evaluating an action's preconditions through the use of conditional expressions. Our proposal combines planning with the application of priority information, improving the reasoning abilities and scope of the planning system presented in [6, 7]. The conditional-preference expressions give the possibility of programming which preference context should be used in each particular situation.

Integrating multi-agent systems, argumentation, and automatic planning is a research topic addressed by several works in the literature [42, 43]. In [42], an argumentation-based model for addressing the collaborative planning problem in teams of agents is presented. The use of argumentation in this approach allows deliberative dialogues facilitating agreements through the interchange of information about collaborative tasks. In contrast, rather than focusing on defining a collaborative planning formalism, our proposal seeks to integrate the preferences into the planning process, indicating the priority criterion upon which the reasoning over action preconditions is based. Note that conditional expressions provide a useful tool to agree on a criterion to adopt by agents when engaged in dialogue.

Recently, [43] presented a planning system based on DeLP [15] 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 can be discussed among agents; thus, the proposed dialog mechanism allows agents to exchange arguments about the conditions that might affect an action's feasibility according to their knowledge and beliefs. The main difference with this work and our own is that they do not specify what information is prioritized by agents. Integrating our proposal to the proposals of [42, 43] would add interesting characteristics to this kind of system. The conditional-preference expressions presented could be useful to model tools that allow the planner to decide what priority criterion should be used by agents participating in a dialog, depending on the available information.

An approach for practical reasoning was proposed in [45], providing grounds for formalizing the relationship between values and actions and integrating defeasible argumentation into the agent reasoning process. In this formalism, the values that an agent holds are used 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. As in the case of our work, their proposal is based on defeasible argumentation and presents a preference-based approach, but there are some differences between the two approaches. Despite sharing the motivation of integrating preferences into an argumentation-based plan construction formalism, they do not adopt a mechanism to change the agent's preferences. To decide how to choose between actions plans, they take into account those values that the agent considers important, whereas we present an algorithm to construct plans. In [45], the authors focus on comparing plans based on a hierarchical order over values.

The work of [12] concerns epistemic planning problems, focusing on an argumentation-based approach. The paper aims to take the first step in developing an approach to handle contextual preferences that can dynamically change based on knowledge-based priorities. They introduce an architecture that is 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, and the authors also present a concrete instantiation based on Possibilistic Defeasible Logic Programming. Here, we have revised, refined, and extended the proposal in [12]. There are therefore several significant differences in our work. We formally define the main concepts employed in [12] to decide which actions to keep and which not should be applied during the construction of plans by considering the conditional-preference expression associated with each action. We also extend this work by introducing a section to discuss interferences that can appear when such expressions are used. Finally, we present an extension of the APOP algorithm [7], as well as the application example developed in Section 8 to show

how a plan is constructed.

Shams *et al.* [46] propose a framework for normative planning problems that use argumentation to reason about goals and norms in order to identify the best course of action. A planner will be able to select the best plan to execute by prioritizing certain goals and norms over others. In particular, they focus on showing how an agent can decide on the justifiability of the plans and use its preferences to resolve the different types of conflicts within and between goals and norms, and identify which plans are the best to follow. Similarly, our proposal presents an argumentation-based approach, but there are some important differences with [46], perhaps the most salient being that our work focuses on structured rather than abstract argumentation. Moreover, although both approaches can prioritize information in the planning process, Shams *et al.* do not focus on the specification of preferences. An interesting feature of our proposal is that [46] could be extended by incorporating the notion of context-adaptable priority selection introduced here as a particular way of deciding between plans.

The need for adding explainability features to planning systems has been recently recognized by many researchers. In [47], an argumentation-based approach is proposed to generate explanations for planning solutions as well as for invalid plans. While this work—as in our case—uses an argumentation formalism to model planning problems, the focus is not on introducing a particular way of preference representation, but instead on providing explanations associated with plans. Additional related efforts in the area of Explainable AI Planning (XAIP) are [48], [49], and [50].

Finally, several works have been developed to integrate preferences in the planning process [51, 52, 53]. Most of these approaches are not argumentation-based and rely on methodologies and issues related to the development of approaches that study how to generate preferred plans and, ideally, those that are optimal. There is no unique way to carry out this task. To address the problem of preference-based planning, most of the research efforts rely on defining a language for specifying users' preferences, as well as a formalism for specifying and reasoning over plans. Most preference languages in the literature have either a quantitative or qualitative nature, while some of them admit a combination of both. Although our proposal is part of the same general effort to include preferences within the planning process, our main aim is not the generation of preferred plans. Here, we focus on defining a way of adjusting priorities on information to be used when plans are being constructed depending on the current state in which the system acts. Our work can therefore be seen as complementing the current research in planning with preferences. Detailed descriptions of many of preference-based planners are discussed by Baier and McIlraith [54]. The interested reader is refereed to there work for further details.

The works discussed in this section are summarized in Table 1. The table briefly describes, for each related work that is particularly close to ours, the strengths, limitations, and aspects that can be enhanced with mechanisms presented here.

Reference	**Contributions**	**Features**	**Poss. Improvements**
Rahwn & Amgoud [41]	A framework based on argumentation to reason about desires and plans.	(*i*) Structured argumentation-based approach. (*ii*) No tools for interchangeable plan preference criteria.	Using our conditional preference expressions to select and change criteria.
García *et al.* [7]	Argumentation-based formalism for building plans.	(i) DeLP-based approach. (ii) Use of actions' effects and arguments for satisfying preconditions. (iii) Algorithm extending POP to consider arguments as plan steps.	By specifying conditional expressions, our proposal can be used for incorporating into actions an easy and natural way to change priorities on information necessary to achieve the planning system's goals.
Pardo *et al.* [8]	A proposal to extend DeLP-POP [7] to a multiagent environment.	(i) Proposal using DeLP as reasoning mechanism. (ii) No mechanism for preference representation is used.	Our conditional preference expressions could be used as a particular way of expressing explicitly specified context-dependent preferences.
Toniolo *et al.* [42]	Argumentation-based approach to deal the colloborative planning problem in teams of agents.	(i) Use argumentative dialogues about norms and actions. (ii) No algorithm to construct plans is specified.	To use our conditional expressions to provide a useful tool to agree on a priority criterion to adopt by agents during the deliberative dialogue.
Pajares-Ferrando & Onaindia [43]	Planning system for cooperative multiagent environments that uses context information when plans are built.	(i) Based on DeLP. (ii) Extend the DeLP-POP formalism [7] to a multiagent scenario. (iii) No priority order over rules is presented. (iv) Provides a specification of the qualification problem.	The agent's preferences can be encoded as prioritized DeLP-rules, and conditional expressions can be used as an easy way to capture context-dependent preferences.
Teze *et al.* [45]	An argumentation-based proposal to formalize the relationship between values and actions.	(i) Based on DeLP. (ii) Agents' values are used to compare plans. (iii) Several strategies defined to compare plans based on the agent's value system. (iv) No mechanism to modify agents' preferences are introduced.	Our approach could be useful as a tool to change values that actions promote depending current state of the world.

Teze & Godo [12]	Argumentation-based formalism proposing a set of software engineering guidelines to analyze and design planning systems with capacities for handling contextual preferences.	(i) Takes the first step in considering preferences that can dynamically change via knowledge-based priorities. (ii) Introduces an architecture that leverages this preference handling capacity.	The intuitions behind this formalism can be revised, refined, and extended with our approach.
Shams *et al.* [46]	Argumentation-based framework for planning problems in normative environments.	(i) Applies abstract argumentation. (ii) Investigates the properties of the best plan(s). (iii) Decides on the best plan to follow using argumentation. (iv) Shows how preferences between goals and norms can be applied to decide on the best plan.	The formalism could be extended incorporating the intuitive notion of context-adaptable priority selection introduced here as a particular way to decide between plans.
Fan [47]	An argumentation approach that make an initial step towards explainable planning.	(i) Uses structured argumentation. (ii) Argumentation as a modelling tool for solving planning problems and deriving explanations from solution plans. (iii) No specific tool to dynamically change preferences is introduced.	(i) Our context-adaptable priorities selection could be exploited to show explanations focused on preferences under which system reasoning is based. (ii) Knowledge-based priorities could offer a means for deciding between plans.

Table 1: A summary of the main aspects of the related work discussion.

10. Conclusion and Future Work

In this paper, we proposed an argumentation-based approach that planning systems can use to construct plans, where each action is executed under specific preferences selected via different priority criteria. To select the priority criterion under which the system's reasoning is based, we proposed to use a particular type of conditional expression, which provides systems with a way to carry out context-adaptable selection. For the construction of plans, we refined and extended the work presented in [12], and proposed an algorithm that extends the APOP algorithm to consider conditional expressions in its implementation. We have also addressed several types of interferences (threats) that can appear and need to be tackled to obtain a valid plan. Our focus was not on developing tools for improving the planning process's efficiency, but rather on introducing an argumentation-based formalism to incorporate the use of preferences in such a process. For defeasible reasoning over system knowledge, we proposed to use the P-DeLP inference mechanism. We have also focused on presenting several complexity results of our approach; to the best of our knowledge, these are the first complexity results in the argumentation-based planning literature.

As future work, there are several lines of promising research under consideration. We wish to develop an implementation of a planner agent based on the proposal presented, and are also interested in studying how to integrate our approach with a collaborative planning context where several points of view (criteria) are considered by agents involved.

Even though the results we obtained show the feasibility of developing our proposed tools, there are several limitations associated with our efforts that point to the need for further developments and evaluations. Here, we have focused particularly on exploring the aspects centered on the theoretical design of the proposed approach. Thus, computational aspects such as running time efficiency, though of great importance for our long-term goal, fall out of the scope of this paper. A line of work that we are interested in involves the implementation of the framework (and its incorporation into the recently-developed P-DAQAP platform [55]), focusing on achieving scalability and an empirical evaluation in a real scenario, comparing it with other approaches from the literature with respect to effectiveness and efficiency measures. In the formalism, the P-APOP algorithm does not benefit from any heuristic, so another promising issue 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. Finally, extending this work to include other preference representation tools—such as operators for combining or prioritizing contexts—is also a challenging objective to be addressed.

As we mentioned previously, the use of argumentation in planning systems is not new. An interesting area of future work is to investigate the relationship between our approach and emerging conflict between plans. For example, we can follow the approach proposed in [41], and include our proposal of conditional expressions; in this way, it would be possible to consider multiple preferences when partial plans are in conflict. An additional line that we also intend to explore in the future is the relationship between the notion of threat and attack present in the argumentation literature. On the other hand, the assignment of values to rules based on some kind of rationality principle like the one proposed in [56] is also a challenging objective we want to explore as future work.

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/N046 and PGI 24/ZN34, CONICET, and Agencia Nacional de Promoción Científica y Tecnológica, Argentina under grant PICT-2018-0475 (PRH-2014-0007), and Universidad Nacional de Entre Ríos (UNER), by PDTS-UNER (grants 7066). Godo acknowledges partial support by the Spanish project ISINC (PID2019-111544GB-C21).

References

[1] T. Bolander, A gentle introduction to epistemic planning: The DEL approach, in: S. Ghosh, R. Ramanujam (Eds.), Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017, Vol. 243 of EPTCS, 2017, pp. 1–22.

[2] C. Baral, T. Bolander, H. van Ditmarsch, S. A. McIlraith, Epistemic planning (dagstuhl seminar 17231), Dagstuhl Reports 7 (6) (2017) 1–47.

[3] T. Bolander, M. Birkegaard Andersen, Epistemic planning for single and multi-agent systems, Journal of Applied Non-Classical Logics 21 (1) (2011) 9–34.

[4] T. J. M. Bench-Capon, P. E. Dunne, Argumentation in artificial intelligence, Artif. Intell. 171 (10-15) (2007) 619–641.

[5] I. Rahwan, G. R. Simari, Argumentation in Artificial Intelligence, 1st Edition, Springer Publishing Company, Incorporated, 2009.

[6] D. R. García, A. J. García, G. R. Simari, Planning and defeasible reasoning, in: 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007, 2007, p. 222.

[7] D. R. García, A. J. García, G. R. Simari, Defeasible reasoning and partial order planning, in: Foundations of Information and Knowledge Systems, 5th International Symposium, FoIKS 2008, Pisa, Italy, February 11-15, 2008, Proceedings, 2008, pp. 311–328.

[8] P. Pardo, S. Pajares, E. Onaindia, L. Godo, P. Dellunde, Multiagent argumentation for cooperative planning in DeLP-POP, in: 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3, 2011, pp. 971–978.

[9] T. C. Son, E. Pontelli, Planning with preferences using logic programming, TPLP 6 (5) (2006) 559–607.

[10] A. Jorge, S. A. McIlraith, et al., Planning with preferences, AI Magazine 29 (4) (2008) 25–25.

[11] L. Bidoux, J. Pignon, F. Bénaben, Planning with preferences using multiattribute utility theory along with a choquet integral, Eng. Appl. Artif. Intell. 85 (2019) 808–817.

[12] J. C. Teze, L. Godo, An architecture for argumentation-based epistemic planning: A first approach with contextual preferences, IEEE Intelligent Systems 36 (2) (2020) 43–51.

[13] J. C. Teze, S. Gottifredi, A. J. García, G. R. Simari, Improving argumentation-based recommender systems through context-adaptable selection criteria, Expert Syst. Appl. 42 (21) (2015) 8243–8258.

[14] L. Amgoud, S. Parsons, L. Perrussel, An argumentation framework based on contextual preferences., in: Proceedings of the 3rd International Conference on Formal and Applied Practical Reasoning, FAPR '00, 2000, 2000, pp. 59–67.

[15] A. J. García, G. R. Simari, Defeasible logic programming: An argumentative approach, Theory and Practice of Logic Programming (TPLP) 4 (1-2) (2004) 95–138.

[16] T. Alsinet, C. I. Chesñevar, L. Godo, G. R. Simari, A logic programming framework for possibilistic argumentation: Formalization and logical properties, Fuzzy Sets and Systems 159 (10) (2008) 1208–1228.

[17] C. I. Chesñevar, G. R. Simari, T. Alsinet, L. Godo, A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge, in: D. M. Chickering, J. Y. Halpern (Eds.), UAI '04, Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence, Canada, AUAI Press, 2004, pp. 76–84.

[18] T. Alsinet, C. I. Chesñevar, L. Godo, S. A. Sandri, G. R. Simari, Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification, International Journal of Approximate Reasoning 48 (3) (2008) 711–729.

[19] T. Alsinet, R. Béjar, L. Godo, F. Guitart, Rp-delp: a weighted defeasible argumentation framework based on a recursive semantics, J. Log. Comput. 26 (4) (2016) 1315–1360.

[20] D. Dubois, J. Lang, H. Prade, Possibilistic Logic, in: D. Gabbay, C. Hogger, J. Robinson, D. Nute (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Nonmonotonic Reasoning, and Uncertain Reasoning, Vol. 3, Claredon Press, Oxford, 1994, pp. 439–513.

[21] S. Benferhat, D. Dubois, H. Prade, Towards a possibilistic logic handling of preferences, Applied Intelligence 14 (2001) 303–317. doi:10.1023/A:1011298804831.

[22] D. Dubois, H. Prade, A bipolar possibilistic representation of knowledge and preferences and its applications, in: I. Bloch, et al. (Eds.), Fuzzy Logic and Applications, 6th International Workshop, WILF 2005, Revised Selected Papers, Vol. 3849 of Lecture Notes in Computer Science, Springer, 2005, pp. 1–10.

[23] S. Kaci, Working with Preferences: Less Is More, Cognitive Technologies, Springer, 2011.

[24] S. Benferhat, D. Dubois, S. Kaci, H. Prade, Possibilistic logic representation of preferences: relating prioritized goals and satisfaction levels expressions, in: F. van Harmelen (Ed.), Proceedings of the 15th European Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002, IOS Press, 2002, pp. 685–689.

[25] D. Dubois, H. Prade, Fuzzy sets in approximate reasoning, part 1: Inference with possibility distributions, Fuzzy Sets Syst. 40 (1) (1991) 143–202.

[26] D. Dubois, H. Prade, The logical view of conditioning and its application to possibility and evidence theories, International Journal of Approximate Reasoning 4 (1990) 23–46.

[27] V. Lifschitz, Foundations of logic programs, in: G. Brewka (Ed.), Principles of Knowledge Representation, Center for the Study of Language and Information, USA, 1997, pp. 69–128.

[28] J. C. Teze, S. Gottifredi, A. J. García, G. R. Simari, An approach to generalizing the handling of preferences in argumentation-based decision-making systems, Knowledge-Based Systems (2019) 105112.

[29] A. Barrett, D. S. Weld, Partial-order planning: Evaluating possible efficiency gains, Artif. Intell. 67 (1) (1994) 71–112.

[30] G. Brassard, P. Bratley, Fundamentals of algorithmics, Prentice-Hall, Inc., 1996.

[31] C. H. Papadimitriou, Computational complexity, Addison-Wesley, 1994.

[32] P. Bachmann, Die analytische zahlentheorie, Vol. 2, Teubner, 1894.

[33] E. Landau, Handbuch der Lehre von der Verteilung der Primzahlen, Teubner, Leipzig, 1909.

[34] M. Y. Vardi, The complexity of relational query languages, in: Proc. STOC, 1982, pp. 137–146.

[35] C. A. Knoblock, Q. Yang, Evaluating the trade-offs in partial-order planning algorithms, Tech. rep., University of Southern California Marina Del Rey Information Sciences Inst (1994).

[36] T. Bylander, Complexity results for planning, in: J. Mylopoulos, R. Reiter (Eds.), Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991, Morgan Kaufmann, 1991, pp. 274–279.

[37] L. A. Cecchi, P. R. Fillottrani, G. R. Simari, On the complexity of DeLP through game semantics, in: Proc. 11th Intl. Workshop on Nonmonotonic Reasoning (NMR 2006), Clausthal University Germany, 2006, pp. 386–394.

[38] C. H. Papadimitriou, M. Yannakakis, The complexity of facets (and some facets of complexity), in: Proceedings of the fourteenth annual ACM symposium on Theory of computing, 1982, pp. 255–260.

[39] G. Alfano, S. Greco, F. Parisi, G. I. Simari, G. R. Simari, Incremental computation for structured argumentation over dynamic DeLP knowledge bases, Artificial Intelligence (2021) 103553.

[40] K. L. Myers, T. J. Lee, Generating qualitatively different plans througt metatheoretic biases, in: Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA., 1999, pp. 570–576.

[41] I. Rahwan, L. Amgoud, An argumentation based approach for practical reasoning, in: AAMAS, 2006, pp. 347–354.

[42] A. Toniolo, T. J. Norman, K. P. Sycara, Argumentation schemes for collaborative planning, in: Agents in Principle, Agents in Practice - 14th International Conference, PRIMA 2011, Wollongong, Australia, November 16-18, 2011. Proceedings, 2011, pp. 323–335.

[43] S. Pajares-Ferrando, E. Onaindia, Defeasible-argumentation-based multi-agent planning, Inf. Sci. 411 (2017) 1–22.

[44] S. Pajares-Ferrando, E. Onaindia, A. Torreño, An architecture for defeasible-reasoning-based cooperative distributed planning, in: R. M. et al. (Ed.), OTM 2011, Proceedings, Part I, Vol. 7044 of Lecture Notes in Computer Science, Springer, 2011, pp. 200–217.

[45] J. C. Teze, A. Perello-Moragues, L. Godo, P. Noriega, Practical reasoning using values: an argumentative approach based on a hierarchy of values, Ann. Math. Artif. Intell. 87 (3) (2019) 293–319.

[46] Z. Shams, M. D. Vos, N. Oren, J. A. Padget, Argumentation-based reasoning about plans, maintenance goals, and norms, ACM Trans. Auton. Adapt. Syst. 14 (3) (2020) 9:1–9:39.

[47] X. Fan, On generating explainable plans with assumption-based argumentation, in: T. Miller, N. Oren, Y. Sakurai, I. Noda, B. T. R. Savarimuthu, T. C. Son (Eds.), PRIMA 2018: Principles and Practice of Multi-Agent Systems - 21st International Conference, Tokyo, Japan, October 29 - November 2, 2018, Proceedings, Vol. 11224 of Lecture Notes in Computer Science, Springer, 2018, pp. 344–361.

[48] N. Oren, K. van Deemter, W. W. Vasconcelos, Argument-based plan explanation, in: M. Vallati, D. E. Kitchin (Eds.), Knowledge Engineering Tools and Techniques for AI Planning, Springer, 2020, pp. 173–188.

[49] Q. Mahesar, S. Parsons, Argument schemes and dialogue for explainable planning, CoRR abs/2101.02648.

[50] L. A. Dennis, N. Oren, Explaining BDI agent behaviour through dialogue, in: F. Dignum, A. Lomuscio, U. Endriss, A. Nowé (Eds.), AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021, ACM, 2021, pp. 429–437.

[51] T. A. Nguyen, M. B. Do, A. Gerevini, I. Serina, B. Srivastava, S. Kambhampati, Generating diverse plans to handle unknown and partially known user preferences, Artif. Intell. 190 (2012) 1–31.

[52] M. Das, P. Odom, M. R. Islam, J. R. Doppa, D. Roth, S. Natarajan, Preference-guided planning: An active elicitation approach, in: E. André, S. Koenig, M. Dastani, G. Sukthankar (Eds.), Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018, International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018, pp. 1921–1923.

[53] J. Fu, Probabilistic planning with preferences over temporal goals, in: 2021 American Control Conference, ACC 2021, New Orleans, LA, USA, May 25-28, 2021, IEEE, 2021, pp. 4854–4859.

[54] J. A. Baier, S. A. McIlraith, Planning with preferences, AI Mag. 29 (4) (2008) 25–36.

[55] M. A. Leiva, A. J. García, P. Shakarian, G. I. Simari , Argumentation-based query answering under uncertainty with application to cybersecurity, Big Data and Cognitive Computing 6 (3). doi:10.3390/bdcc6030091.

[56] A. Hunter, A probabilistic approach to modelling uncertain logical arguments, Int. J. Approx. Reason. 54 (1) (2013) 47–81.