A temporal argumentation approach to cooperative planning using dialogues

PERE PARDO, *Department of Philosophy and Logic and Philosophy of Science, Universidad de Sevilla, Av. Camilo José Cela s/n, Sevilla 41018, Spain. E-mail: ppardo1@us.es*

LLUÍS GODO, *Artificial Intelligence Research Institute (IIIA - CSIC), Campus UAB s/n, Bellaterra 08193, Spain. E-mail: godo@iiia.csic.es*

Abstract

In this paper, we study a dialogue-based approach to multi-agent collaborative plan search in the framework of t-DeLP, an extension of DeLP for defeasible temporal reasoning. In t-DeLP programs, temporal facts and rules combine into arguments, which compare against each other to decide which of their conclusions are to prevail. By adding temporal actions for multiple agents to this argumentative logic programming framework, one obtains a centralised planning framework. In this planning system, it can be shown that breadth-first search is sound and complete for both forward and backward planning. The main contribution is to extend these results in centralised planning to cooperative planning tasks, where the executing agents themselves are assumed to have reasoning and planning abilities. In particular, we propose a planning algorithm where agents exchange information on plans using suitable dialogues. We show that the soundness and completeness properties of centralised t-DeLP plan search are preserved, so the dialoguing agents will reach an agreement upon a joint plan if and only if some solution exists.

Keywords: Planning, Argumentation, Multi-Agent Systems, Defeasible Logic, Temporal Reasoning, Dialogue

1 Introduction

The relatively recent area of argumentation [6] has become a focus of attention in AI and multi-agent systems. Argumentation systems provide (human-inspired) frameworks upon which agents can resolve conflicts between different claims or arguments.

In particular, abstract argumentation tools have been combined with different logics, giving rise to new non-monotonic reasoning systems for inconsistency handling, called logic-based argumentation. These are systems that aim to model commonsense, causal or evidential reasoning (e.g. birds fly, solids fall, etc.), as well as reasoning about actions. In either case, a practical need exists for general-purpose criteria of preference that select among conflicting inferences. This need led to the formal notion of specificity [23], and later to argumentation systems applying this idea to define the attack relation, e.g. in the defeasible logic programming DeLP system [9]. According to this criterion, a preference relation can be defined between arguments, by formally comparing their logical structure or informational content. Both the DeLP system and the criteria of specificity have been recently adapted to temporal reasoning, where

arguments contain temporal information explicitly [2, 5, 15]. A common motivation for these extensions seems to be common-sense causal reasoning. Finally, several planning systems built upon these logics have been studied, e.g. partial-order planning DeLP-POP [7, 8, 19], and forward planners for the temporal argumentative t-DeLP system [17]. The latter is an adaptation of the DeLP system to a temporal rule-based language, with minor differences due to specific issues in temporal reasoning [15].

Among the motivations to build planners on some of these non-monotonic logical systems we find the classical AI representation problems. Some of these issues seriously affect traditional planners, since their reasoning about actions (i.e. their action update) is limited by simple monotonic inference and the assumption that the consequences of an action can be encapsulated as a list of direct or conditional effects. Despite the simplicity attained in the resulting state transition systems, these planning systems prove insufficient to deal with classical representation problems, broadly known as the frame problem. In this sense, while these systems need not to costly compute all the facts that do not change after an action (thus avoiding the narrow frame problem), they cannot address the problems of modelling the indirect effects of actions (the ramification problem), or qualifications on their preconditions either (the qualification problem). Clearly, adding argumentative capabilities to a planning system does increase the computational cost of the system. However, the use of defeasible information will permit us to deal with planning problems which cannot be modeled with traditional approaches.

Another recent topic of interest in the areas of logic and automated planning is that of multi-agent approaches to traditional problems. Along the line of recent trends in these areas, the main aim of this paper is the study of a distributed multi-agent planning system, particularly based on the t-DeLP framework for temporal argumentation. This system combines a traditional update function (for temporal actions) with non-monotonic inference based on the t-DeLP notion of warrant, or logical consequence. For a multi-agent approach, we study distributed plan search algorithms for cooperative scenarios. Towards this end, we present first a centralised algorithm for the t-DeLP planning system, where each executing agent is assigned its contribution to a joint plan built by an external planner. This approach prevents one from considering more interesting scenarios with autonomous agents, where each agent can plan, reason and communicate with the other agents. The aim of the latter is to avoid centralised solutions, which might incur in an important and unnecessary loss in terms of information privacy and autonomy.

To this end, we extend centralised planning into cooperative planning domains, where each agent is initially endowed with a planning domain of its own, including beliefs, abilities and (a shared set of) goals. In the present paper, the initial planning domain of an agent particularly consists of a t-DeLP logic program (beliefs), a set of temporal actions (abilities) and a set of temporal literals representing the common goals. Instead of a central planner, then, we propose a dialogue-based algorithm for distributed plan search in the t-DeLP planning system. In the proposed dialogues, all agents take part in the generation of a joint plan by communicating (only) information which seems relevant to the present task. The dialogues consist in each agent taking turns to address the next agent in line. All the agents contribute to the generation or evaluation of new plan steps for plans already under consideration. In this cooperative planning framework, the main results are soundness and completeness theorems for

the dialogue-based plan search algorithm. Thus, setting complexity issues aside, a group of cooperative agents are as good as a central planner can be to generate a joint plan, if the latter is assumed to have access to all of their information.

This paper, which is an extended and revised version of two conference papers [17, 18], is structured as follows. In Section 2 we briefly review first the t-DeLP temporal defeasible logic programming framework. Then in Section 3, we adapt the basic concepts of planning systems to the case of t-DeLP, including an appropriate update function for t-DeLP and the notions of planning domain, plan and solution, and study the soundness and completeness of a centralised breadth-first search algorithm for backward plan search in this planning system. In Section 4 we propose a distributed algorithm (also in the spirit of breadth-first search) for the multi-agent t-DeLP planning system based on a dialogue between agents; and finally we show that this algorithm is also sound and complete. To improve the readability of this paper, the proofs of the main theorems can be found in an Appendix. See the manuscript [16] for more details.

Notation. We make use of the following conventions. Set-theoretic difference between two sets X, Y is denoted $X \setminus Y$. Sequences are denoted $\langle x_0, \ldots, x_n \rangle$ (general case) or $[x_0, \ldots, x_n]$ (for argumentation lines) or $(x_0, \ldots, x_n)$ (for plans). Given a sequence $\vec{x} = \langle x_0, \ldots, x_n \rangle$ and an element x, we denote by $\vec{x}^{\frown}\langle x \rangle$ the concatenation of $\vec{x}$ with x, i.e. the sequence $\langle x_0, \ldots, x_n, x \rangle$ or $[x_0, \ldots, x_n, x]$. If f is a function $f : X \to Y$ and $X' \subseteq X$, we define $f[X'] = \{f(a) \in Y \mid a \in X'\}$.

2 Preliminaries: Temporal Defeasible Logic Programming

The t-DeLP temporal logic programming framework, a temporal adaptation of the DeLP system from [8], is briefly reviewed in the present section, see also [15, 16]. Later the t-DeLP system is used as the logical foundation of the planning system studied in the next sections. We will use throughout the paper Example 2.1 as a running example to illustrate both the argumentative and planning concepts to be introduced; see also Figure 1 for an illustration of this example.

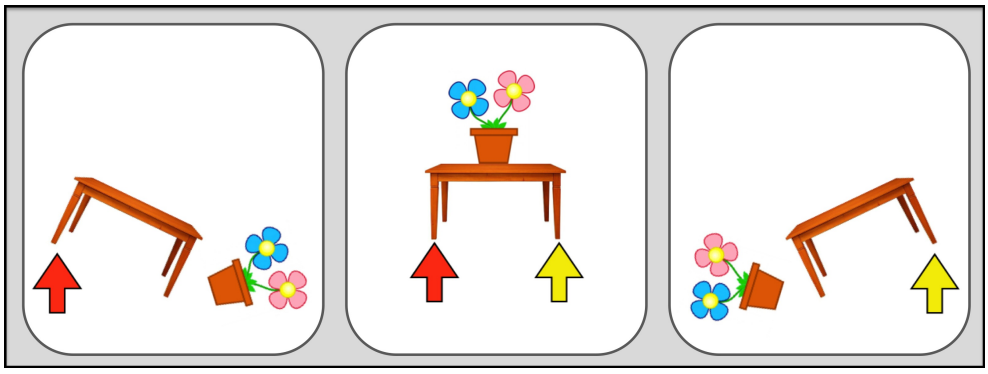

FIG. 1: (Left) and (Right) The vase breaks if only one side of the table is lifted. (Center) The vase stays on the table if both sides are lifted simultaneously.

EXAMPLE 2.1 (Lifting a Table –Running Example)
Consider a table which can be lifted on each of its two sides (north and south), and assume there is a vase on the table. Lifting only one side of the table causes the vase to fall off and break, as depicted in Figure 1 (Left) and (Right). Lifting both sides of the table simultaneously, in contrast, causes the vase to stay on the table, see Fig. 1 (Center).

The language of t-DeLP consists of a set of *temporal rules* built upon a set of *temporal literals*. Temporal literals are pairs of the form $\langle \ell, t\rangle$, read as ℓ *holds at time* t, where ℓ is a literal p or $\sim p$ from a given set of variables $p \in \mathsf{Var}$, and t is a time point. We consider discrete time, so t will take values in the set $\mathbb{N}$ of natural numbers. For strong negation $\sim$, since $\sim\sim p \equiv p$, we adopt the following notation over literals: if $\ell = p$ then $\sim\ell$ will denote $\sim p$, and if $\ell = \sim p$ then $\sim\ell$ will denote p. Time determines if a pair of temporal literals contradict each other; namely, this pair must be of the form $\langle \ell, t\rangle, \langle \sim\ell, t\rangle$. The set of temporal literals is denoted $\mathsf{TLit} = \mathsf{Var} \times \mathbb{N}$.

DEFINITION 2.2 (t-DeLP language [15])
Given a finite set of atoms Var a *temporal rule* is an expression δ of the form

$$\begin{array}{ll} \text{defeasible} & \langle \ell, t\rangle \prec \langle \ell_0, t_0\rangle, \ldots, \langle \ell_n, t_n\rangle \\ \text{strict} & \langle \ell, t\rangle \leftarrow \langle \ell_0, t_0\rangle, \ldots, \langle \ell_n, t_n\rangle \end{array} \quad \text{where } t \geq \max\{t_0, \ldots t_n\}$$

We let $\mathsf{body}(\delta)$ denote the set of conditions $\{\langle \ell_0, t_0\rangle, \ldots, \langle \ell_n, t_n\rangle\}$ and let $\mathsf{head}(\delta)$ denote its conclusion $\langle \ell, t\rangle$. A *(strict) fact* is a strict rule with an empty body $\langle \ell, t\rangle \leftarrow$. A strict fact will also be denoted simply as $\langle \ell, t\rangle$.

EXAMPLE 2.3 (Lifting a Table –language)
The above Example 2.1 (without actions) will be formalized using the next set of atoms Var:

$$\begin{array}{llll} b = \mathsf{broken(vase)} & h = \mathsf{horizontal(table)} & \mu_{\mathsf{N}} = \mu_{\mathsf{lift.N}} & l_{\mathsf{N}} = \mathsf{lifted_N} \\ f = \mathsf{falls.off(vase)} & o = \mathsf{on(vase, table)} & \mu_{\mathsf{S}} = \mu_{\mathsf{lift.S}} & l_{\mathsf{S}} = \mathsf{lifted_S} \end{array}$$

The variable μ_{N} (resp. μ_{S}) represents the fact that *the action "lift north" was executed* (resp. *"lift south"*); see Section 3.1.

A defeasible rule δ states that if the premises in $\mathsf{body}(\delta)$ are true, then there is a reason for believing that the conclusion $\mathsf{head}(\delta)$ is also true. This conclusion, though, may be later withdrawn when further information is considered. In contrast, strict rules necessarily preserve the truth from the body to the head of a rule. In practice, we will restrict the set of strict rules to those induced by mutex sets –see below.

In order to avoid time paradoxes, t-DeLP only makes use of future-oriented rules, so $\mathsf{head}(\delta)$ cannot occur earlier than any $\langle \ell_k, t_k\rangle \in \mathsf{body}(\delta)$. A special subset of defeasible rules is that of *persistence* rules, of the form $\langle \ell, t+1\rangle \prec \langle \ell, t\rangle$, stating that ℓ is preserved from t to $t+1$. Such a rule will denoted as $\delta_\ell(t)$.

EXAMPLE 2.4 (Lifting a Table –strict facts; defeasible rules)
Let us model the above scenario (without actions), using the *strict facts*: $\langle \sim b, 0\rangle$, $\langle o, 0\rangle$, $\langle h, 0\rangle, \langle \sim l_{\mathsf{N}}, 0\rangle$, $\langle \sim l_{\mathsf{S}}, 0\rangle$. These temporal literals state that, at initial time $t = 0$, *the vase is unbroken and on the table, and the table is horizontal with none of its sides being currently lifted*. In addition, we will use the next defeasible rules for common-sense causal reasoning:

$\delta_1 : \langle \sim h, t\rangle \ \multimap \langle l_{\mathsf{N}}, t\rangle$ $\quad$ $\delta_2 : \langle \sim h, t\rangle \ \multimap \langle l_{\mathsf{S}}, t\rangle$

$\delta_3 : \langle h, t\rangle \ \multimap \langle l_{\mathsf{N}}, t\rangle, \langle l_{\mathsf{S}}, t\rangle$ $\quad$ $\delta_4 : \langle l_{\mathsf{N}}, t+1\rangle \ \multimap \langle \mu_{\mathsf{N}}, t+1\rangle, \langle \sim l_{\mathsf{N}}, t\rangle$

$\delta_5 : \langle l_{\mathsf{S}}, t+1\rangle \ \multimap \langle \mu_{\mathsf{S}}, t+1\rangle, \langle \sim l_{\mathsf{S}}, t\rangle$ $\quad$ $\delta_6 : \langle b, t\rangle \ \multimap \langle f, t\rangle$

$\delta_7 : \langle f, t+1\rangle \ \multimap \langle \sim h, t\rangle, \langle o, t\rangle$ $\quad$ $\delta_8 : \langle \sim o, t\rangle \ \multimap \langle f, t\rangle$

$\delta_\ell(t)$: persistence rules for each literal $\ell \in \{\sim b, o, l_{\mathsf{N}}, l_{\mathsf{S}}, \sim l_{\mathsf{N}}, \sim l_{\mathsf{S}}\}$ and each $t < 10$.

Definition 2.5 (Derivation; consistency [15])
Given a set of temporal rules and literals Γ, we say a literal $\langle \ell, t\rangle$ *derives from* Γ, denoted $\Gamma \vdash \langle \ell, t\rangle$, or also $\langle \ell, t\rangle \in \mathrm{Cn}(\Gamma)$, iff $\langle \ell, t\rangle \in \Gamma$ or there exists $\delta \in \Gamma$ with $\mathsf{head}(\delta) = \langle \ell, t\rangle$, and such that $\mathsf{body}(\delta)$ is a set of literals that derive from Γ. We say Γ is *consistent* iff no pair $\langle \ell, t\rangle, \langle \sim \ell, t\rangle$ exists in $\mathrm{Cn}(\Gamma)$.

In particular, a set of literals is consistent iff it contains no contradictory pair $\langle \ell, t\rangle, \langle \sim \ell, t\rangle$. Note that derivability is monotonic: $\mathrm{Cn}(\Gamma) \subseteq \mathrm{Cn}(\Gamma')$ whenever $\Gamma \subseteq \Gamma'$.

Definition 2.6 (Program; Mutex program)
A *(t-DeLP) program* is a pair (Π, Δ) where $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathsf{r}}$ is a consistent set of facts (Π_{f}) and strict rules (Π_{r}), and Δ is a set of temporal defeasible rules. The set of persistence rules, denoted Δ_{p}, is a subset of Δ.

A *mutex set* X is a subset of variables $X \subseteq \mathsf{Var}$; a *mutex family* $\mathbf{M}$ is a collection of mutex sets $\mathbf{M} = \{X, \ldots\}$. A program (Π, Δ) is called *mutex* iff Π_{r} is of the form

$$\Pi_{\mathsf{r}} = \bigcup_{X \in \mathbf{M}} \{\langle \sim q, t\rangle \leftarrow \langle p, t\rangle \mid p, q \in X \text{ and } t \in \mathbb{N}\}$$

for some mutex family $\mathbf{M}$.

Intuitively, a pair in a mutex set $p, q \in X \in \mathbf{M}$ expresses a mutual exclusion between facts p and q (at any given time t), so pairs of the form $\langle p, t\rangle, \langle q, t\rangle$ cannot be accepted, as we do with inconsistent pairs $\langle q, t\rangle, \langle \sim q, t\rangle$. To this end, the mutex rule $\langle \sim q, t\rangle \leftarrow \langle p, t\rangle$ permits to derive the latter inconsistency from the pair $\langle p, t\rangle, \langle q, t\rangle$.

Example 2.7 (Mutex rules for physical constraints)
Let $@(o, l)$ be an atom denoting that *object o is at location l*. The set $X_o = \{@(o, l), @(o, l'), \ldots\}$ will induce mutex rules $\langle \sim @(o, l'), t\rangle \leftarrow \langle @(o, l), t\rangle$, etc., forbidding o to be at different places $l \neq l'$ at the same time t. Similarly, the set $X_l = \{@(o, l), @(o', l), \ldots\}$ can prevent that any two objects $o, o', \ldots$ exist at the same place and time. The mutex family $\mathbf{M} = \{X_o, X_{o'}, \ldots, X_l, X_{l'}, \ldots\}$ would include both kinds of constraints for all objects and locations.

Example 2.8 (Lifting a Table –program)
The facts and rules from Example 2.4, together with the effects $\langle \mu_{\mathsf{N}}, t\rangle, \langle \mu_{\mathsf{S}}, t\rangle$ (for some unspecified time $0 < t < 10$) can be gathered into a program (Π, Δ) for the scenario where both sides of the table become lifted at time t:

$$\begin{aligned} \Pi &= \{\langle \sim b, 0\rangle,\ \langle h, 0\rangle,\ \langle \sim l_{\mathsf{N}}, 0\rangle,\ \langle \sim l_{\mathsf{S}}, 0\rangle,\ \langle o, 0\rangle,\ \langle \mu_{\mathsf{N}}, t\rangle,\ \langle \mu_{\mathsf{S}}, t\rangle\} \\ \Delta &= \{\delta_1, \ldots, \delta_8\}_{0 \le t \le 10} \cup \{\delta_\ell(t)\}_{0 \le t < 10}\ . \end{aligned}$$

The rules and facts of a program can combine into arguments for further derived facts. Informally, an argument for a conclusion is a minimal set of facts and rules that is consistent with the strict part Π, from which the conclusion is derivable.

DEFINITION 2.9 (Argument [15])
Given a program (Π, Δ), an *argument* for $\langle \ell, t\rangle$ is a set $\mathcal{A} = \mathcal{A}_\Pi \cup \mathcal{A}_\Delta$, with $\mathcal{A}_\Pi \subseteq \Pi$ and $\mathcal{A}_\Delta \subseteq \Delta$, such that:

(1) $\mathcal{A}_\Delta \cup \Pi \vdash \langle \ell, t\rangle$,
(2) $\Pi \cup \mathcal{A}_\Delta$ is consistent,
(3) $\mathcal{A}_\Delta$ is $\subseteq$-minimal satisfying (1) and (2).
(4) $\mathcal{A}_\Pi$ is $\subseteq$-minimal satisfying $\mathcal{A}_\Delta \cup \mathcal{A}_\Pi \vdash \langle \ell, t\rangle$

Given an argument $\mathcal{A}$ for $\langle \ell, t\rangle$, we also define $\mathsf{concl}(\mathcal{A}) = \langle \ell, t\rangle$, and $\mathsf{base}(\mathcal{A}) = \bigcup \mathsf{body}[\mathcal{A}] \setminus \mathsf{head}[\mathcal{A} \setminus \Pi_\mathsf{f}]$ and $\mathsf{literals}(\mathcal{A}) = (\bigcup \mathsf{body}[\mathcal{A}]) \cup \mathsf{head}[\mathcal{A}]$. [1] [2]

EXAMPLE 2.10 (Lifting a Table –argument)
If the table is lifted on both sides at time t, in (Π, Δ) one can build arguments for the following conclusions: *the table remains horizontal at time t* ($\mathcal{A}_3$); *the table does not remain horizontal at t* ($\mathcal{B}_1^-, \mathcal{B}_2^-$); *the vase breaks at* $t+1$ ($\mathcal{B}_1, \mathcal{B}_2$); and *the vase remains unbroken at time 10* ($\mathcal{A}_1$); see also Fig. 2(Left).

argument	*rules and facts*	*conclusion*
$\mathcal{A}_1$	$\{\delta_{\sim b}(t')\}_{0\le t'<10} \cup \{\langle \sim b, 0\rangle\}$	$\langle \sim b, 10\rangle$
$\mathcal{A}_3$	$\{\delta_3, \delta_4, \delta_5, \langle \mu_\mathsf{N}, t\rangle, \langle \mu_\mathsf{S}, t\rangle, \langle \sim l_\mathsf{N}, 0\rangle, \langle \sim l_\mathsf{S}, 0\rangle\} \cup \cup \{\delta_{l_\mathsf{N}}(t'), \delta_{l_\mathsf{S}}(t')\}_{0\le t'<t}$	$\langle h, t\rangle$
$\mathcal{B}_1^-$	$\{\delta_1, \delta_4\} \cup \{\delta_{l_\mathsf{N}}(t')\}_{0\le t'<t} \cup \{\langle \mu_\mathsf{N}, t\rangle, \langle \sim l_\mathsf{N}, 0\rangle\}$	$\langle \sim h, t\rangle$
$\mathcal{B}_1$	$\mathcal{B}_1^- \cup \{\delta_o(t')\}_{0\le t'<t} \cup \{\delta_6, \delta_7, \langle o, 0\rangle\}$	$\langle b, t+1\rangle$
$\mathcal{B}_2^-$	$\{\delta_2, \delta_5\} \cup \{\delta_{l_\mathsf{S}}(t')\}_{0\le t'<t} \cup \{\langle \mu_\mathsf{S}, t\rangle, \langle \sim l_\mathsf{S}, 0\rangle\}$	$\langle \sim h, t\rangle$
$\mathcal{B}_2$	$\mathcal{B}_2^- \cup \{\delta_o(t')\}_{0\le t'<t} \cup \{\delta_6, \delta_7, \langle o, 0\rangle\}$	$\langle b, t+1\rangle$

In the next definition, it can be shown that the sub-argument induced by a literal is indeed unique. We also use the notation $\sim\langle \ell, t\rangle$ as another way to express $\langle \sim\ell, t\rangle$.

DEFINITION 2.11 (Sub-argument; Attack [15])
Given an argument $\mathcal{A}$ in a program (Π, Δ), and a literal $\langle \ell_0, t_0\rangle \in \mathsf{literals}(\mathcal{A})$, we define the *sub-argument* $\mathcal{A}(\langle \ell_0, t_0\rangle)$, as an argument for $\langle \ell_0, t_0\rangle$ such that $\mathcal{A}(\langle \ell_0, t_0\rangle) \subseteq \mathcal{A}$.

Given two arguments, $\mathcal{A}_0, \mathcal{A}_1$ in (Π, Δ), we say that $\mathcal{A}_1$ *attacks* $\mathcal{A}_0$ iff $\sim\mathsf{concl}(\mathcal{A}_1) \in \mathsf{literals}(\mathcal{A}_0)$. In this case, $\mathcal{A}_1$ attacks $\mathcal{A}_0$ at the sub-argument $\mathcal{A}_0(\sim\mathsf{concl}(\mathcal{A}_1))$.

EXAMPLE 2.12 (Lifting a table –subargument, attack)
Example 2.10 contains sub-arguments $\mathcal{B}_1^- \subseteq \mathcal{B}_1$ and $\mathcal{B}_2^- \subseteq \mathcal{B}_2$, and attacks:

$\mathcal{A}_3$	attacks	$\mathcal{B}_1^-, \mathcal{B}_1$	(at $\mathcal{B}_1^-$)		$\mathcal{A}_3$	attacks	$\mathcal{B}_2^-, \mathcal{B}_2$	(at $\mathcal{B}_2^-$)
$\mathcal{B}_1^-, \mathcal{B}_2^-$	attack	$\mathcal{A}_3$	(at $\mathcal{A}_3$)		$\mathcal{B}_1, \mathcal{B}_2$	attack	$\mathcal{A}_1$ (at $\mathcal{A}_1(\langle \sim b, t+1\rangle)$)	

Since attacks among arguments are symmetrical (the attacked sub-argument also attacks the attacking argument), we need a more refined notion (defeat) to enforce some sort of asymmetry. This notion of *defeat* is established by comparing an attacking argument and the attacked (sub-)argument. The comparison is made in terms of their use of strict facts (more is better) and their use of persistence rules (less is better).

[1] Note that condition (3) requires $\mathcal{A}$ to make a minimal use of defeasible rules (i.e. to use strict rules if available), for which a simpler alternative condition (1') $\mathcal{A}_\Delta \cup \mathcal{A}_\Pi \vdash \langle \ell, t\rangle$ in place of (1) above would not suffice.

[2] Let us also note that a simpler definition $\mathsf{base}(\mathcal{A}) = \Pi_\mathsf{f} \cap \bigcup \mathsf{body}[\mathcal{A}]$ suffices for logical reasoning. The more general definition given here suits better the needs of backward planning.

DEFINITION 2.13 (Defeat [16])
Given a mutex program (Π, Δ) and two arguments $\mathcal{A}_0, \mathcal{A}_1$ in (Π, Δ), let $\mathcal{A}_1$ attack $\mathcal{A}_0$ at a sub-argument $\mathcal{B}$, where $\mathsf{concl}(\mathcal{A}_1) = \langle \sim\ell, t\rangle$. We say that $\mathcal{A}_1$ is a *proper defeater for* $\mathcal{A}_0$, denoted $\mathcal{A}_1 \succ \mathcal{A}_0$, iff

$$\mathsf{base}(\mathcal{A}_1) \not\supseteq \mathsf{base}(\mathcal{B}) \qquad \text{or} \qquad \begin{array}{c}\mathcal{B} \cap \mathcal{A}_1 \text{ is an argument for some } \langle \ell', t'\rangle \text{ with } t' < t \\ \text{and } \mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} \text{ (so } \ell' = \ell \text{ or } \ell, \ell' \in X \in \mathbf{M})\end{array}$$

We say $\mathcal{A}_1$ is a *blocking defeater for* $\mathcal{A}_0$ when $\mathcal{A}_1$ attacks $\mathcal{A}_0$ but $\mathcal{A}_1 \not\succ \mathcal{A}_0$ and $\mathcal{A}_0 \not\succ \mathcal{A}_1$. Blocking defeat relations are denoted $\mathcal{A}_1 \prec\!\succ \mathcal{A}_0$. Finally, a *defeater* is a proper or a blocking defeater.[3]

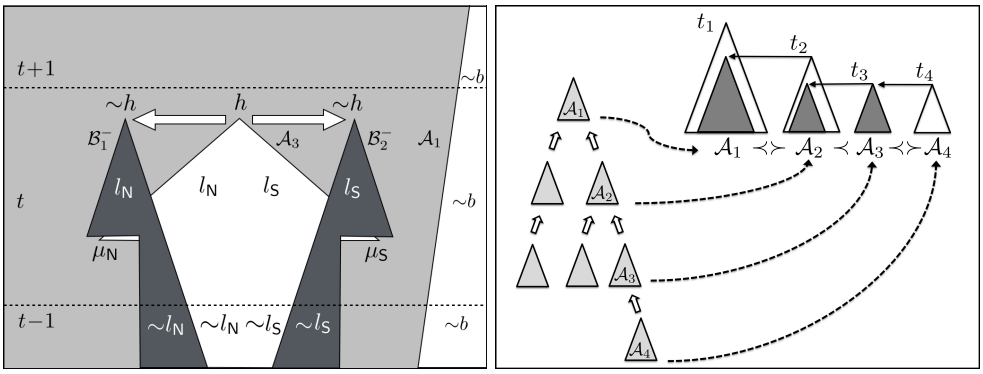

FIG. 2: (Left) A depiction of the arguments $\mathcal{A}_1, \mathcal{A}_3, \mathcal{B}_1^-, \mathcal{B}_2^-$ from Example 2.10, in the case where both sides of the table are simultaneously lifted at t. The Y-axis represents the time instants associated to facts, rules and conclusions in these arguments. The figure depicts the $[t-1, t+1]$ fragments of the arguments $\mathcal{A}_1$, $\mathcal{B}_1^-$, $\mathcal{B}_2^-$ and $\mathcal{A}_3$. Note that $\mathcal{A}_3$ defeats $\mathcal{B}_1^-$ and $\mathcal{B}_2^-$ –as well as their extensions $\mathcal{B}_1, \mathcal{B}_2$ (not depicted). (Right) An argumentation line $\Lambda = [\mathcal{A}_1, \ldots, \mathcal{A}_4]$ in the dialectical tree for $\mathcal{A}_1$ in some program (Π, Δ); defeated sub-arguments are depicted in grey. The time of these attacks is (non-strictly) decreasing: $t_1 > t_2 > t_3 = t_4$.

EXAMPLE 2.14 (Lifting a Table –defeat)
In Example 2.10, there are some proper defeats: $\mathcal{A}_3 \succ \{\mathcal{B}_1, \mathcal{B}_1^-, \mathcal{B}_2, \mathcal{B}_2^-\}$, as well as *blocking defeats* $\{\mathcal{B}_1, \mathcal{B}_2\} \prec\!\succ \mathcal{A}_1$. Note that the weak case of $\mathcal{B}_1, \mathcal{B}_2$ against $\mathcal{A}_1$ is strongly challenged by argument $\mathcal{A}_3$.

As this example shows, an argument $\mathcal{B}_1$ defeating $\mathcal{A}_1$ can in its turn have its own defeaters $\mathcal{A}_3$ and so on. This gives rise to an *argumentation line* for the argument $\mathcal{A}_1$, defined next –see also Fig. 2(Right).

DEFINITION 2.15 (Argumentation line [15])
An *argumentation line* for $\mathcal{A}_1$ is a sequence of arguments $\Lambda = [\mathcal{A}_1, \ldots, \mathcal{A}_n]$ such that:

(i) supporting arguments, i.e. those in odd positions $\mathcal{A}_{2i+1} \in \Lambda$ are jointly consistent with Π, and similarly for interfering arguments $\mathcal{A}_{2i} \in \Lambda$.

[3] In [2], the authors propose an alternative criterion for persistence, where an argument containing persistence rules is to be defeated (in case of attack) by any other argument without persistence rules.

(ii) a supporting (interfering) argument is different from the attacked sub-arguments of previous supporting (interfering) arguments: $\mathcal{A}_{i+2k} \neq \mathcal{A}_i(\sim\mathsf{concl}(\mathcal{A}_{i+1}))$.

(iii) $\mathcal{A}_{i+1}$ is a proper defeater for $\mathcal{A}_i$ if $\mathcal{A}_i$ is a blocking defeater for $\mathcal{A}_{i-1}$. [4]

The set of maximal argumentation lines for $\mathcal{A}_1$ can be arranged in the form of a tree, where all paths $[\mathcal{A}_1, \ldots]$ exactly correspond to all the possible maximal argumentation lines for $\mathcal{A}_1$. This *dialectical tree* for $\mathcal{A}_1$ is denoted $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. See Figure 2 (Right) for an illustration of the dialectical tree for $\mathcal{A}_1$ and its argumentation lines. In order to evaluate an argument $\mathcal{A}_1$ we apply the next procedure on its dialectical tree.

Definition 2.16 (Marking procedure [9])
The marking procedure in a dialectical tree $\mathcal{T} = \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ is:

(1) mark all terminal nodes of $\mathcal{T}$ with a U (for undefeated);

(2) mark a node $\mathcal{B}$ with a D (for defeated) if it has a child node marked U;

(3) mark $\mathcal{B}$ with U if all its child nodes are marked D.

Definition 2.17 (Warrant [9])
Given a program (Π, Δ), we say $\langle \ell, t\rangle$ is *warranted* in (Π, Δ) if there exists an argument $\mathcal{A}$ for $\langle \ell, t\rangle$ in (Π, Δ) such that $\mathcal{A}$ is marked undefeated (U) in the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. The set of warranted literals is denoted $\mathsf{warr}(\Pi, \Delta)$.

Example 2.18 (Lifting a Table –warrant)
The arguments from Example 2.10 can be arranged into the following argumentation lines $\{[\mathcal{A}_1, \mathcal{B}_1, \mathcal{A}_3], [\mathcal{A}_1, \mathcal{B}_2, \mathcal{A}_3]\} = \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. Computing warrant with this dialectical tree gives $\{[U, D, U], [U, D, U]\}$, so $\mathcal{A}_1$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ and its conclusion is warranted, i.e. $\langle \sim b, 10\rangle \in \mathsf{warr}(\Pi, \Delta)$. In summary, the vase remains not broken at time 10 if both sides of the table are lifted simultaneously at some $t < 10$.

One can show that t-DeLP enjoys the next logical properties, called Rationality Postulates [4, 20], for the class of mutex programs. These postulates prevent certain types of counter-intuitive results.

Theorem 2.19 (Postulates for argumentation)
The following properties hold for arbitrary t-DeLP mutex programs (Π, Δ):

- *Sub-arguments*: if $\mathcal{A}$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$, then any sub-argument $\mathcal{A}'$ of $\mathcal{A}$ is also undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}')$.
- *Direct Consistency*: $\mathsf{warr}(\Pi, \Delta)$ is consistent.
- *Indirect Consistency*: $\mathsf{warr}(\Pi, \Delta) \cup \Pi$ is consistent.
- *Closure*: $\mathrm{Cn}(\mathsf{warr}(\Pi, \Delta) \cup \Pi) \subseteq \mathsf{warr}(\Pi, \Delta)$

We refer the reader to [16] for a proof of this theorem.

[4] Some discussion exists within the *defeasible argumentation* community about what definition of *argumentation line* would be most appropriate or intuitively correct. For example, H. Prakken suggests to adopt the argumentation game for grounded semantics: the proponent of an argument can only use proper defeaters (i.e. for the supporting arguments), while the opponent can use both proper and blocking defeaters (for the interfering arguments).With appropriate modifications of proofs, this and other reasonable definitions should preserve Theorem 2.19 below.

3 A centralised Planning System for t-DeLP

After this brief presentation of t-DeLP, we proceed to introduce a planning system based on the t-DeLP temporal argumentation framework. First we adapt the basic concepts of action and update to the present framework, in order to define a t-DeLP-based state transition system. Here, a state will correspond to (the warr-closure of) a t-DeLP program at a given time-point. Next, we introduce a multi-agent planning system built on this system and define plans built under backward search. Finally, we review backward search algorithms for centralised planning in this t-DeLP planning system. Throughout the current section, then, a central planner will be tasked with the construction of a joint plan for multiple executing agents, and assign them the corresponding tasks.

As mentioned at the start of this paper, a motivation for an approach combining argumentation with action update are the classical representation problems, e.g. the ramification problem. The indirect or contextual effects of an action play an important role when the scenario already includes actions executed by other agents.

Example 3.1 (Lifting a Table –adding actions)
Recall the example from Figure 1, where one might have as initial facts either that only the north side of the table is lifted $\langle \mu_{\mathsf{N}}, t \rangle$, or only its south side $\langle \mu_{\mathsf{S}}, t \rangle$, or both –as in Example 2.8. For the purpose of planning, from here on the μ-facts can only obtain after updating the program with actions. Thus, we will introduce two lifting actions lift.N or lift.S, and redefine the program (Π, Δ) (from Example 2.8) with the following set

$$\Pi = \{\langle \sim b, 0 \rangle,\ \langle o, 0 \rangle,\ \langle h, 0 \rangle, \langle \sim l_{\mathsf{N}}, 0 \rangle,\ \langle \sim l_{\mathsf{S}}, 0 \rangle\}$$

An advantage of using t-DeLP to compute causal effects is that, in e.g. the previous example, these two atomic actions lift.N, lift.S can be combined to describe the three events from Fig. 1. In contrast, classical planning would demand to add either a new "atomic" action, say lift.N&S, or two conditional effects for each action that depend upon the (non-)occurrence of the other action. Instead of this, we will define a unique set of defeasible rules to correctly compute the causal effects of any combination of these two atomic actions. To do so, we need to adapt the notion of action update from classical planning to work together with t-DeLP reasoning.

3.1 Update of t-DeLP programs with temporal actions

As usual in planning, an action is just a pair of consistent sets (*preconditions*, *post-conditions*), but for the purpose of this paper, several assumptions are made on the representation of actions in order to simplify the description of the planning system. Thus, each action e in the set of actions A can only be executed by an agent, has a duration of 1 time unit, and has a unique effect $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle$. This effect is read as *action* e *was just executed (at time* t_{e}*)*, and we assume that the language contains one such literal for each action e described. Moreover, the effect $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle$ is exclusive to this action (not found in nature or other actions) and it cannot be contradicted once it is made true (by other rules or arguments). We also assume that the execution of an action e by an agent a makes this agent busy during the execution. Finally, for the purpose of this paper, we will assume the existence of (a finite but) sufficient number of agents for the problem at hand.

In what follows we will assume a t-DeLP language has been fixed. Let us proceed with the basic definitions of action and update.

DEFINITION 3.2 (Action; Executability; Non-overlapping actions)
An *action* is a pair $\mathsf{e} = \mathsf{e}^t = (\mathsf{pre}(\mathsf{e}), \mathsf{post}(\mathsf{e}))$, where $\mathsf{pre}(\mathsf{e}) = \{\langle \ell, t\rangle, \ldots, \langle \ell', t\rangle\}$ is a consistent set of temporal literals and $\mathsf{post}(\mathsf{e}) = \{\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle\}$, with $t_{\mathsf{e}} = t + 1$. These are called the *preconditions* and the *(direct) effect* of e. [5]

An action e is *executable* in a program (Π, Δ) iff $\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\Pi, \Delta)$. Given a set of agents $\mathsf{Ag} = \{a, b, \ldots\}$, a set of actions A is *non-overlapping w.r.t.* Ag iff for any two actions in A of the same agent a, say e, f, the effect of e is to occur strictly before the preconditions of f, or viceversa.

EXAMPLE 3.3 (Lifting a table –actions)
For Example 3.1, consider a set of two agents $\mathsf{Ag} = \{a_1, a_2\}$. The actions $\mathsf{lift.N}^9, \mathsf{lift.S}^9$ of lifting either side of the table at $t = 9$ can be defined as the following pairs:

$$\begin{array}{llll} \mathsf{pre}(\mathsf{lift.N}^9) & = \{\langle @(a_1, \mathsf{N}), 9\rangle\} & \mathsf{pre}(\mathsf{lift.S}^9) & = \{\langle @(a_2, \mathsf{S}), 9\rangle\} \\ \mathsf{post}(\mathsf{lift.N}^9) & = \{\langle \mu_{\mathsf{N}}, 10\rangle\} & \mathsf{post}(\mathsf{lift.S}^9) & = \{\langle \mu_{\mathsf{S}}, 10\rangle\} \end{array}$$

In practice, though, we will assume that both agents are at the right place, so we can forget about these preconditions and set $\mathsf{pre}(\mathsf{lift.N}^t) = \varnothing = \mathsf{pre}(\mathsf{lift.S}^t)$.

DEFINITION 3.4 (Action Update)
The *update* of a program (Π, Δ) by an action e, denoted $(\Pi, \Delta) \diamond \mathsf{e}$, is defined as:

$$(\Pi, \Delta) \diamond \mathsf{e} = \begin{cases} (\Pi \cup \mathsf{post}(\mathsf{e}), \Delta) & \text{if } \mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\Pi, \Delta) \\ (\Pi, \Delta) & \text{otherwise.} \end{cases}$$

Since actions are assigned an execution time by their preconditions, any set of actions $\{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$ can be seen as being implicitly ordered. Indeed, the order of execution of two simultaneous actions does not matter.

LEMMA 3.5 ([16])
For any program (Π, Δ) and literals $\langle \ell, t\rangle, \langle \ell', t'\rangle$ with $t < t'$, the following hold:

$$\langle \ell, t\rangle \in \mathsf{warr}(\Pi, \Delta) \quad \Leftrightarrow \quad \langle \ell, t\rangle \in \mathsf{warr}(\Pi \cup \{\langle \ell', t'\rangle\}, \Delta).$$

If $\mathsf{e}_i, \mathsf{e}_j$ are simultaneous actions with $\mathsf{pre}(\mathsf{e}_i) = \{\langle \ell, t\rangle, \ldots\}$ and $\mathsf{pre}(\mathsf{e}_j) = \{\langle \ell', t\rangle, \ldots\}$,

$$((\Pi, \Delta) \diamond \mathsf{e}_i) \diamond \mathsf{e}_j = ((\Pi, \Delta) \diamond \mathsf{e}_j) \diamond \mathsf{e}_i.$$

The previous Lemma enables the following definition.

DEFINITION 3.6 (Iterated Update)
The *update* of a program (Π, Δ) by a set of actions is recursively defined as follows:

$$(\Pi, \Delta) \diamond \varnothing \quad = \quad (\Pi, \Delta)$$

and if $A = \{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$ is a set of actions e_k with $\mathsf{pre}(\mathsf{e}_k) = \{\langle \ell_k, t_k\rangle, \ldots\}$, then

$$(\Pi, \Delta) \diamond A \quad = \quad ((\Pi, \Delta) \diamond \mathsf{e}_i) \diamond (A \smallsetminus \{\mathsf{e}_i\})$$

[5] For notational simplicity in proofs, we work directly with a set A of (temporally instantiated) actions, rather than operators –as usually done in planning. The conversion into the latter form is immediate and, moreover, each operator would stand for finitely-many action instances, since goals (temporal literals) have deadlines.

where e_i is any action in A whose execution time t_i is minimal among $\{t_1, \ldots, t_n\}$.

EXAMPLE 3.7 (Lifting a Table –action update)
Recall the program (Π, Δ) from Example 2.8, and the actions $\mathsf{lift.N}^9$ and $\mathsf{lift.S}^9$ from Example 3.3. Then, the updated program $(\Pi, \Delta) \diamond \{\mathsf{lift.N}^9, \mathsf{lift.S}^9\}$ gives rise to exactly the same arguments and warranted conclusions from Example 2.8 or Fig. 2(Left).

3.2 A temporal planning system based on t-DeLP

After presenting the state transition system defined by t-DeLP programs and update with temporal actions, one can easily extend this system into a planning system by adapting the definitions of planning domain and solution to the t-DeLP framework.

DEFINITION 3.8 (Planning Domain)
Given a set of agents Ag, we define a *planning domain* as a triple

$$\mathbb{M} = (\mathbb{P}, A, G)$$

where $\mathbb{P} = (\Pi, \Delta)$ is a t-DeLP program representing the domain knowledge,[6] with the facts in Π representing the *initial state*; A is a set of *actions* available to the agents in Ag and G is a set of literals expressing the *goals*.

A planner is to find a set of actions that make the goals warranted after update.

DEFINITION 3.9 (Solution)
Given a set of agents Ag and a planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$, a set of actions $A' \subseteq A$ is a *solution* for $\mathbb{M}$ iff

$$G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A') \quad \text{and} \quad A' \text{ is non-overlapping w.r.t. } \mathsf{Ag}.$$

EXAMPLE 3.10 (Lifting a Table –planning domain; solution)
Let us suppose that the planner, endowed with two agents $\mathsf{Ag} = \{a_1, a_2\}$, wants the table to be lifted at $t = 10$, without breaking the vase which lies on it. We define a planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$, where (Π, Δ) is the t-DeLP program from Example 2.8, and the goals G and actions A are described as follows:

$$G = \{\langle l_\mathsf{N}, 10\rangle, \langle l_\mathsf{S}, 10\rangle, \langle \sim b, 10\rangle\} \qquad A = \{\mathsf{lift.N}^t, \mathsf{lift.S}^t\}_{0<t<10}$$

Consider the set of actions $A' = \{\mathsf{lift.N}^9, \mathsf{lift.S}^9\}$ or, more generally, any set of the form $A' = \{\mathsf{lift.N}^t, \mathsf{lift.S}^t\}$ with $0 < t < 10$. We will obtain two undefeated arguments $\mathcal{A}_4$ and $\mathcal{A}_5$ resp. for the goals $\langle l_\mathsf{N}, 10\rangle$ and $\langle l_\mathsf{S}, 10\rangle$.

$$\mathcal{A}_4 = \{\delta_{l_\mathsf{N}}(t')\}_{t \le t' < 10} \cup \{\delta_4, \langle \mu_\mathsf{N}, t\rangle, \langle \sim l_\mathsf{N}, t-1\rangle\} \cup \{\delta_{\sim l_\mathsf{N}}(t'')\}_{0 \le t'' < t}$$
$$\mathcal{A}_5 = \{\delta_{l_\mathsf{S}}(t')\}_{t \le t' < 10} \cup \{\delta_5, \langle \mu_\mathsf{S}, t\rangle, \langle \sim l_\mathsf{S}, t-1\rangle\} \cup \{\delta_{\sim l_\mathsf{S}}(t'')\}_{0 \le t'' < t}$$

These, together with the arguments $\mathcal{A}_1, \mathcal{A}_3$ (from Example 2.10) making $\langle \sim b, 10\rangle$ warranted, suffice to show that A' is a solution for the planning domain $((\Pi, \Delta), A, G)$:

$$G = \{\langle \sim b, 10\rangle, \langle l_\mathsf{N}, 10\rangle, \langle l_\mathsf{S}, 10\rangle\} \subseteq \mathsf{warr}((\Pi, \Delta) \diamond \{\mathsf{lift.N}^t, \mathsf{lift.S}^t\}).$$

[6] The language of (Π, Δ) is assumed to contain a literal μ_e for each action $\mathsf{e} \in A$. Moreover, temporal literals $\langle \mu_\mathsf{e}, t_\mathsf{e}\rangle$ can only occur in the body of the rules of Δ, while those of the form $\langle \sim \mu_\mathsf{e}, t_\mathsf{e}\rangle$ cannot occur anywhere in Π, Δ, A or G.

The idea of t-DeLP backward planning for finding a solution to a given planning domain $\mathbb{M} = (\mathbb{P}, A, G)$ can be sketched as follows: each open goal is to be solved not directly by an action, but as the conclusion of some argument, together with the actions (and initial facts) needed to support it. Then, the planner is to iteratively enforce an undefeated status for this argument, by defeating its defeaters (called threats) whenever these occur, and solve as well the preconditions of the actions supporting it. All this is again done with further arguments, and the corresponding actions. This plan construction terminates when all the goals in G are solved, and all those arguments and actions are, respectively, undefeated and executable. The two types of refinement steps are called *argument steps* –if they solve open goals–, or *threat resolution moves* –if they defeat some threat. Related to this procedure, Figure 3 depicts: (1) a goal, (2) an argument step $\mathcal{A}$ solving it, (3) a threat $\mathcal{B}$ (an interfering argument) triggered by action e, and (4) a threat resolution move $\mathcal{C}$ against $\mathcal{B}$.

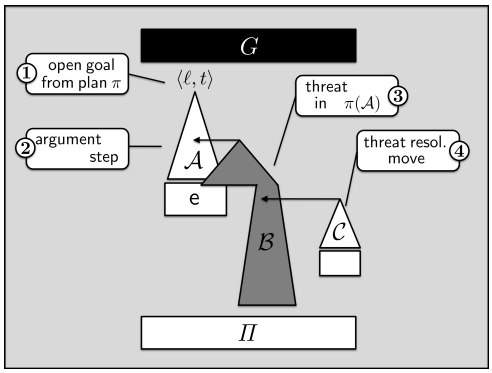

FIG. 3: Steps in the construction of a plan: arguments are triangle-like figures, while actions and facts are represented by rectangles; if an action e is below an argument $\mathcal{A}$, the action supports it, i.e. $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle \in \mathsf{base}(\mathcal{A})$.

In centralised planning, a plan for some planning domain $(\mathbb{P}, A, G)$ is a triple $\pi = (\mathsf{actions}, \mathsf{Trees}, \mathsf{goals})$, where $\mathsf{actions}$ is a set of actions to be executed, Trees is a set of dialectical (sub-)trees (one for each argument step) and goals is the set of open goals of π. For convenience, though, we will denote the triple defining a plan π rather as

$$(A(\pi), \mathsf{Trees}(\pi), \mathsf{goals}(\pi))$$

Thus, the set $\mathsf{Trees}(\pi)$ is used to keep track of argument steps, threats and threat resolution moves. As usual in backward planning, during the construction one abstracts from the full executability of the actions $A(\pi)$ in the initial program $\mathbb{P} = (\Pi, \Delta)$. Hence, rather than considering an update $\mathbb{P} \diamond A(\pi)$, this plan π will induces a new program $\mathbb{P} \oplus \pi = (\Pi \cup \mathsf{post}[A(\pi)], \Delta)$ based on the assumption of executability. In this way, the plan also induces a *provisional* dialectical tree $\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A})$ for each existing argument and, in particular, for each argument step $\mathcal{A}$. In order to avoid unnecessary threat resolution moves, the policy of the planner will be to address each threat $\mathcal{B}$ with a *single* defeater $\mathcal{A}_{k+1}$ for this $\mathcal{B}$. This results in a sub-tree $\mathcal{T}^{*}_{\mathbb{P}\oplus\pi}(\mathcal{A})$ of $\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A})$, to be stored in $\mathsf{Trees}(\pi)$ and later expanded in further refinements of π. See Figure 4 for an illustration.

DEFINITION 3.11 (Plan)
Given a planning domain $\mathbb{M} = (\mathbb{P}, A, G)$, a plan π for $\mathbb{M}$ is obtained from the empty plan $\pi_\varnothing$ after a finite number of refinements with plan steps $\Lambda_1, \ldots, \Lambda_n$, and will be denoted $\pi = \pi_\varnothing(\Lambda_1, \ldots, \Lambda_n)$. A plan step Λ_k is an odd-length tuple of arguments $\Lambda_k = [\ldots, \mathcal{A}_k]$. There are two types of plan steps:

- an *argument step* is a plan step of length one, i.e. of the form $\Lambda_k = [\mathcal{A}_k]$,
- a *threat resolution move* is a plan step of the form $\Lambda_k = [\mathcal{A}_j, \mathcal{B}_j, \ldots, \mathcal{A}_{k-1}, \mathcal{B}_{k-1}, \mathcal{A}_k]$, for some $j < k$, where $[\mathcal{A}_j, \mathcal{B}_j, \ldots, \mathcal{A}_{k-1}, \mathcal{B}_{k-1}]$ is a threat, i.e. an even length argumentation line in the dialectical tree $\mathcal{T}^*_{(\Pi,\Delta)}(\mathcal{A}_j)$.

See Table 1 for a formal definition of plan refinement with argument steps and threat resolution moves. A refinement of a plan π with Λ_{n+1} is denoted $\pi(\Lambda_{n+1})$. Finally, let us denote with $\pi_k = \pi_\varnothing(\Lambda_1, \ldots, \Lambda_k)$, for $1 \leq k \leq n$, the initial fragment of π. Then, the set of *previously solved goals* is defined as

$$\mathsf{OldGoals}(\pi) = \bigcup_{0 \leq k < n} \big(\mathsf{goals}(\pi_k) \setminus \bigcup_{k < k' \leq n} \mathsf{goals}(\pi_{k'})\big)$$

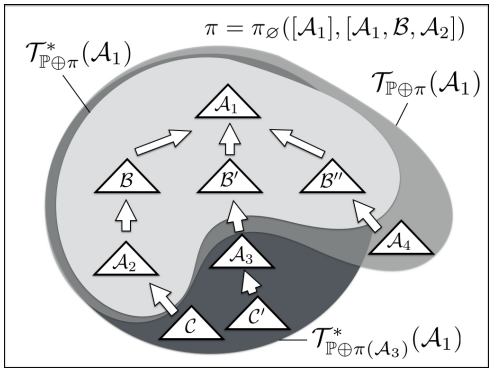

FIG. 4: The dialectical tree for $\mathcal{A}_1$ before and after refining a plan π with $\mathcal{A}_3$. The light grey area represents the provisional tree for $\mathcal{A}_1$ in π, which is a sub-tree of the full dialectical tree (grey area); the latter contains an unplanned, unused threat resolution move $\mathcal{A}_4$. After refining π with $\mathcal{A}_3$, this argument and new threats $\mathcal{C}, \mathcal{C}'$ appear in the new provisional tree after update, represented by the dark grey area.

EXAMPLE 3.12 (Update of $\mathsf{Trees}(\pi)$ after a refinement)
Figure 4 illustrates a refinement of a plan π with a threat resolution move. The new argument $\mathcal{A}_3$ solves an existing threat $[\mathcal{A}_1, \mathcal{B}']$. This causes new threats $\mathcal{C}, \mathcal{C}'$ in the (updated) dialectical tree $\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A}_3)}(\mathcal{A}_1)$. The new plan step $\mathcal{A}_3$ can also cause new threats to other argument steps in π different from $\mathcal{A}_1$, not depicted in Fig. 4.

EXAMPLE 3.13 (Lifting a Table –plan construction)
Recall Example 3.10 describing solutions of the form $A' = \{\mathsf{lift.N}^t, \mathsf{lift.S}^t\}$, for some $0 < t < 10$. Now a planner can build the arguments from Examples 2.10 and 3.10 as argument steps (namely $\mathcal{A}_1, \mathcal{A}_4, \mathcal{A}_5$), threats (e.g. $\mathcal{B}_1, \mathcal{B}_2$) and a threat resolution move (e.g. $\mathcal{A}_3$); for example, consider the successive plan refinements:

TABLE 1. Definition of empty plan, argument step and threat resolution move.

conditions	*definition*
	the **empty plan** for $(\mathbb{P}, A, G)$ is $\pi_\varnothing = (\varnothing, \varnothing, G)$
given a plan $\pi = \pi_\varnothing(\Lambda_1, \ldots, \Lambda_n)$ with $\pi = (A(\pi), \mathsf{Trees}(\pi), \mathsf{goals}(\pi))$ and some $\mathcal{A}^- \subseteq \Pi_\mathsf{r} \cup \Delta$ and $A^* \subseteq A$ satisfying: (i) $A(\pi) \cup A^*$ is non-overlapping (ii) $(\mathbb{P} \oplus \pi) \oplus A^*$ is a program (iii) $\mathcal{A} = \mathcal{A}^- \cup \mathsf{base}(\mathcal{A}^-)$ is an argument w.r.t. (ii) and $\mathsf{concl}(\mathcal{A}) \in \mathsf{goals}(\pi)$ (iv) A^* is $\subseteq$-minimal with (iii) (v) $\mathsf{pre}[A^*]$ is consistent with all literals in arg. steps and all solved/open goals	an **argument step** refinement is a triple $\pi(\mathcal{A}) = (A(\pi(\mathcal{A})), \mathsf{Trees}(\pi(\mathcal{A})), \mathsf{goals}(\pi(\mathcal{A})))$ defined by: • $A(\pi(\mathcal{A})) = A(\pi) \cup A^*$ • $\mathsf{goals}(\pi(\mathcal{A})) = (\mathsf{goals}(\pi) \cup \mathsf{pre}[A^*]) \setminus$ $\setminus(\{\mathsf{concl}(\mathcal{A})\} \cup \Pi_\mathsf{f} \cup \mathsf{OldGoals}(\pi))$ • $\mathsf{Trees}(\pi(\mathcal{A}))$ is a set containing $\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A}) =$ $\{[\mathcal{A}]\} \cup \{[\mathcal{A}, \mathcal{B}] \mid [\mathcal{A}, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}\oplus(\pi(\mathcal{A}))}(\mathcal{A})\}$ $\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A}_k) = \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}_k) \cup$ $\cup \{\Lambda_m{}^\frown[\mathcal{B}] \in \mathcal{T}_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A}_k)\}_{1\leq m\leq n}$ for each argument step $\Lambda_k = [\mathcal{A}_k]$ in π
given a plan $\pi = \pi_\varnothing(\Lambda_1, \ldots, \Lambda_n)$ with $\pi = (A(\pi), \mathsf{Trees}(\pi), \mathsf{goals}(\pi))$ and a threat $\Lambda' = [\mathcal{A}, \ldots, \mathcal{B}] \in \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A})$, let $\mathcal{C}^- \subseteq \Pi_\mathsf{r} \cup \Delta$ and $A^* \subseteq A$ satisfying: (i) $A(\pi) \cup A^*$ is non-overlapping (ii) $(\mathbb{P} \oplus \pi) \oplus A^*$ is a program (iii) $\mathcal{C} = \mathcal{C}^- \cup \mathsf{base}(\mathcal{C}^-)$ is an argument and $\Lambda = \Lambda'^\frown[\mathcal{C}]$ an argumentation line in the program $(\mathbb{P} \oplus \pi) \oplus A^*$ (iv) A^* is $\subseteq$-minimal with (iii)	a **threat resolution move** is a triple $\pi(\Lambda) = (A(\pi(\Lambda)), \mathsf{Trees}(\pi(\Lambda)), \mathsf{goals}(\pi(\Lambda)))$ defined by: • $A(\pi(\Lambda)) = A(\pi) \cup A^*$ • $\mathsf{goals}(\pi(\Lambda)) = (\mathsf{goals}(\pi) \cup \mathsf{pre}[A^*]) \setminus$ $\setminus(\Pi_\mathsf{f} \cup \mathsf{OldGoals}(\pi))$ • $\mathsf{Trees}(\pi(\Lambda))$ containing for each $\mathcal{A}_k \neq \mathcal{A}$ $\mathcal{T}^*_{\mathbb{P}\oplus\pi(\Lambda)}(\mathcal{A}_k) = \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}_k) \cup$ $\cup \{\Lambda_m{}^\frown[\mathcal{B}'] \in \mathcal{T}_{\mathbb{P}\oplus\pi(\Lambda)}(\mathcal{A}_k)\}_{1\leq m\leq n}$ $\mathcal{T}^*_{\mathbb{P}\oplus\pi(\Lambda)}(\mathcal{A}) = \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}) \cup \{\Lambda'^\frown[\mathcal{C}]\} \cup$ $\cup\{\Lambda'^\frown[\mathcal{C}, \mathcal{B}'] \mid \Lambda'^\frown[\mathcal{C}, \mathcal{B}'] \in \mathcal{T}_{\mathbb{P}\oplus(\pi(\Lambda))}(\mathcal{A})\}$

$\pi_\varnothing$	open goals G; no threats
$\pi_\varnothing(\mathcal{A}_1)$	solves goal $\langle{\sim}b, 10\rangle$; no goals are added
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4)$	solves goal $\langle l_\mathsf{N}, 10\rangle$; new threat $[\mathcal{A}_1, \mathcal{B}_1]$
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4, \mathcal{A}_5)$	solves goal $\langle l_\mathsf{S}, 10\rangle$; new threat $[\mathcal{A}_1, \mathcal{B}_2]$
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4, \mathcal{A}_5, [\mathcal{A}_1, \mathcal{B}_1, \mathcal{A}_3])$	solves $[\mathcal{A}_1, \mathcal{B}_1]$
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4, \mathcal{A}_5, [\mathcal{A}_1, \mathcal{B}_1, \mathcal{A}_3], [\mathcal{A}_1, \mathcal{B}_2, \mathcal{A}_3])$	solves $[\mathcal{A}_1, \mathcal{B}_2]$; terminating condition.

Note that the latter plan, call it π, involves the same actions as in the solution described, i.e. $A(\pi) = \{\mathsf{lift.N}^t, \mathsf{lift.S}^t\} = A'$, so π encodes a solution $A(\pi)$ for $\mathbb{M}$.

In contrast, Figure 5 (Right) describes the case where the two sides are not lifted at the same time. Here, the threat $\mathcal{B}_1$ cannot be solved once it occurs in plans involving actions of the form $\{\mathsf{lift.N}^t, \mathsf{lift.S}^{t'}\}$ for some $t \neq t'$. These plans do not end up in solution plans. In summary, the goals $\langle{\sim}b, 10\rangle$, $\langle l_\mathsf{N}, 10\rangle$ and $\langle l_\mathsf{S}, 10\rangle$ are warranted iff the two agents lift both sides simultaneously, as in Fig. 5 (Left).

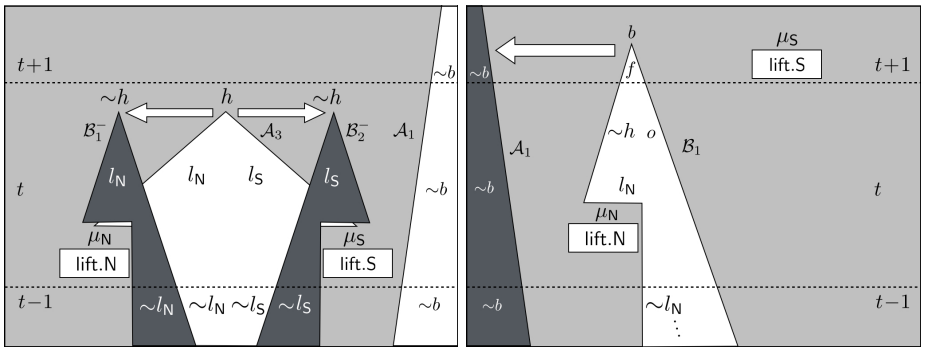

FIG. 5: (Left) An illustration of the arguments after update with both lifting actions at t from Example 3.10. The proper and blocking defeaters are the same as in Example 2.14. (Right) An illustration of update with non-simultaneous lifting actions (at t, and resp. at $t+1$); here, the argument $\mathcal{A}_1$ for *the vase will not break* is ultimately defeated by the threat $\mathcal{B}_1$; i.e. the dialectical tree for $\mathcal{A}_1$ becomes $\{[\mathcal{A}_1, \mathcal{B}_1]\}$.

From here on, if no confusion occurs, a plan $\pi_\varnothing(\Lambda_1, \ldots, \Lambda_n)$ will also be denoted by the arguments contributed to at each refinement. For example, $\pi_\varnothing([\mathcal{A}], [\mathcal{A}, \mathcal{B}, \mathcal{C}])$ can simply be denoted as $\pi_\varnothing(\mathcal{A}, \mathcal{C})$.

3.3 Algorithms for t-DeLP backward plan search

The *space of plans* for a planning domain $\mathbb{M}$ is the graph given by the set of plans for $\mathbb{M}$ (in Definition 3.11) and the "*is a refinement of*" relation. In this search space, breadth-first search is instantiated by the following algorithm for backward planning:

Algorithm 1: BFS for backward planning in the t-DeLP planning system

Data: $\mathbb{M} = ((\Pi, \Delta), A, G)$
Result: π (i.e. the set of actions $A(\pi)$); or fail, if $\mathsf{Plans} = \varnothing$
initialization: $\mathsf{Plans} = \langle \pi_\varnothing \rangle$ and $\pi = \pi_\varnothing$;
while $\mathsf{goals}(\pi) \neq \varnothing$ *or* $\mathsf{threats}(\pi) \neq \varnothing$ **do**
  delete π from Plans;
  set $\mathsf{Plans} = \mathsf{Plans}^\frown\langle\, \pi(\mathcal{A}) \mid \pi(\mathcal{A})$ is a refinement of $\pi\rangle$;
  set π = the first element of Plans;
end

This and other usual search methods (depth-first search, etc.) are sound and complete for backward or forward t-DeLP planning. See [17] for more details, and the Appendix for proofs of the next results for breadth-first search (BFS).

THEOREM 3.14 (Soundness of breadth-first plan search)
Let π be an output of the BFS Algorithm 1 in the space of plans for $\mathbb{M}$. Then $A(\pi)$ is a solution for $\mathbb{M}$.

THEOREM 3.15 (Completeness of breadth-first plan search)
Let $\mathbb{M} = ((\Pi, \Delta), A, G)$ be a planning domain and assume some solution $A' \subseteq A$

exists. Then, the BFS search by Algorithm 1 in the space of plans terminates with an output π.

On the other hand, if no solution exists for a planning domain $\mathbb{M}$, the algorithm will terminate with fail after exploring the finitely-many possible sequences of plan steps.

4 Dialogues for distributed planning with cooperative agents

After presenting centralised plan search in the t-DeLP planning system, now we can introduce the dialogue-based planning algorithm for distributed plan search. The first novelty is that now each agent is endowed with a planning domain, although their goals are still the same –the agents are cooperative. The algorithm for an agent determines how she will contribute during her next turn in the dialogue, by supplying the next agent with information for new plan steps and threats, or for claims against the existence of those arguments proposed by other agents. Thus, at each round an agent's planning domain can be expanded. See Figure 6 (Left) for an illustration.

The main contributions of this section are soundness and completeness results for the proposed dialogue-based planning. These results are shown by comparing an agent's solution plan (after the dialogue) to that which would be obtained by a central planner endowed with all of the agents' initial information. In both approaches, the planner agents essentially make use only of the planning methods from Section 3.

4.1 Distributed and centralised planning domains

We first introduce multiple-planner versions of the definitions found in Section 3.2. As we said, each agent $a \in \mathsf{Ag}$ is endowed with an initial planning domain $\mathbb{M}_a$. A sequence of such planning domains $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$, with shared goals $G_a = G$, generates a dialogue for the proposal and discussion of plans.

Definition 4.1 (Multi-planner domain; centralised planning domain)
Given a t-DeLP language and a set of planner agents $\mathsf{Ag} = \{a_1, \ldots, a_r\}$, let $\mathbb{M}_a = ((\Pi_a, \Delta_a), A_a, G)$ be a planning domain for each agent $a = a_i \in \mathsf{Ag}$. Then, we say $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$ is a *multi-planner domain*, if $\bigcup_{a \in \mathsf{Ag}} \Pi_a$ is consistent. We also define the component-wise *union* of two planning domains, say $\mathbb{M}_a, \mathbb{M}_b$, as follows

$$\mathbb{M}_a \sqcup \mathbb{M}_b = ((\Pi_a \cup \Pi_b, \Delta_a \cup \Delta_b), A_a \cup A_b, G)$$

More generally, we define the *centralised planning domain* induced by $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$, denoted $\mathbb{M}_{\mathsf{Ag}}$, as the union of the elements of this multi-planner domain:

$$\mathbb{M}_{\mathsf{Ag}} \;=\; \bigsqcup_{a \in \mathsf{Ag}} \mathbb{M}_a \;.$$

For the sake of simplicity, from here on we keep the notation $\mathsf{Ag} = \{a_1, \ldots, a_n\}$ for the running example, but we may also use instead $\mathsf{Ag} = \{1, \ldots, n\}$ for some definitions and results.

Example 4.2 (Lifting a Table –multi-planner domain)
We rewrite the problem $\mathbb{M}$ of Example 3.10 as a multi-planner domain with $\mathsf{Ag} = \{a_1, a_2\}$. In $\mathbb{M}_{a_1}, \mathbb{M}_{a_2}$, the goals, initial facts and rules are as in $\mathbb{M}$ for both agents,

except that now $\delta_7 \notin \Delta_{a_2}$, so a_2 ignores that *objects on non-horizontal surfaces tend to fall off*. With more detail, for each agent $a \in \mathsf{Ag}$, $\mathbb{M}_a = ((\Pi_a, \Delta_a), A_a, G)$ is defined as

$$\Pi_{a_1} = \Pi = \Pi_{a_2} \qquad \Delta_{a_1} = \Delta \qquad \Delta_{a_2} = \Delta \setminus \{\delta_7\}$$
$$A_{a_1} = \{\mathsf{lift}.\mathsf{N}^t\}_{0<t<10} \qquad A_{a_2} = \{\mathsf{lift}.\mathsf{S}^t\}_{0<t<10} \qquad G = \{\langle l_{\mathsf{N}}, 10\rangle,\ \langle l_{\mathsf{S}}, 10\rangle,\ \langle \sim b, 10\rangle\}$$

The centralised planning domain is simply $\mathbb{M}_{\mathsf{Ag}} = \mathbb{M}_{a_1} \sqcup \mathbb{M}_{a_2} = \mathbb{M}$.

As we said, the plan search methods for a single planner from Section 3 can be used to compute plans in both individual and centralised planning domains $\mathbb{M}_a$ and $\mathbb{M}_{\mathsf{Ag}}$. Indeed, upon receiving a message during a dialogue for cooperative planning, an agent will expand her current planning domain with atomic data (facts, rules or actions) extracted from this message. After this expansion, the agent can compute new ideas for (or against) plans and threats.

Definition 4.3 (Expansion)
Let $\mathbb{M} = ((\Pi, \Delta), A, G)$ and $\mathbb{M}' = ((\Pi', \Delta'), A', G)$ be planning domains. We say $\mathbb{M}'$ is an *expansion* of $\mathbb{M}$, denoted $\mathbb{M} \sqsubseteq \mathbb{M}'$, iff $\Pi \subseteq \Pi'$, $\Delta \subseteq \Delta'$ and $A \subseteq A'$.

Notice in particular that for any pair $\mathbb{M}_1, \mathbb{M}_2$ we have that $\mathbb{M}_1, \mathbb{M}_2 \sqsubseteq \mathbb{M}_1 \sqcup \mathbb{M}_2$. The most important difference with the case of a central planner is that planning domains are no longer static, due to initial differences among agents' initial planning domains. In fact, the agents need not agree on the following:

- whether a given plan step $\mathcal{A}$ exists, i.e. whether $\pi(\mathcal{A})$ actually defines a plan, or
- which plan does such $\pi(\mathcal{A})$ define, i.e. which threats exist, or open goals remain

The source of disagreements about a suggested plan $\pi = \pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ lies in the fact that this sequence π gives rise to different triples (*actions, trees, goals*) when interpreted from different agents' planning domains. For this reason, from here on we introduce a superscript notation for interpreted plans $\pi^{\mathbb{M}}$ and distinguish between

- a plan as a sequence of plan steps $\pi = \pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$, and
- an *interpreted* plan $\pi^{\mathbb{M}}$, denoting the particular result of computing the sequence π in a planning domain $\mathbb{M}$ (only defined if π is actually a plan for $\mathbb{M}$), denoted

$$\pi^{\mathbb{M}} = (A(\pi^{\mathbb{M}}), \mathsf{Trees}(\pi^{\mathbb{M}}), \mathsf{goals}(\pi^{\mathbb{M}})).$$

Note that this is simply Definition 3.11 but with a superindex making explicit which planning domain is being used. Finally, we also add to these concepts from Section 3.2 a new kind of interpretation:

- a *free interpretation* of a plan π in a planning domain $\mathbb{M}$, denoted $\pi^{(\mathbb{M})^+}$; this is an instrumental step towards an interpreted plan $\pi^{\mathbb{M}}$ within a dialogue. Again it is a triple of the form

$$\pi^{(\mathbb{M})^+} = (A(\pi^{\mathbb{M}}), \mathsf{Trees}(\pi^{(\mathbb{M})^+}), \mathsf{goals}(\pi^{\mathbb{M}})),$$

  where $\mathsf{Trees}(\pi^{(\mathbb{M})^+})$ also contains potential (incomplete) threats and claims against the validity of the last plan step or some proposed threats.

4.2 Turn-based Dialogues for Cooperative Planning in t-DeLP.

The dialogues will consist in a series of rounds, each agent speaking once each round, and always to the same agent; see Figure 6 (Left). Starting with turn 1 and agent $a_1 \in \mathsf{Ag}$, the agent speaking at a turn $m > 1$ is $a_{f(m)}$, where $f(m)$ is simply m modulo the number of agents, i.e. $f(m) \equiv m \pmod{|\mathsf{Ag}|}$. The speaking agent $a_{f(m)}$ then communicates a tuple $\mathsf{turn}(m)$ to the agent next in line $a_{f(m+1)}$, of the form (see also Figure 6 (Right))

$$\mathsf{turn}(m) = (\,\mathsf{Preplans}_m,\, \mathsf{Plans}_m,\, \mathsf{Trueplans}_m,\, \mathsf{strict}_m\,).$$

For the planning domain of agent $f(m)$ at turn m, say $\mathbb{M}^m = (\mathbb{P}^m, A^m, G)$, these components informally contain:

- $\mathsf{Preplans}_m =$ the set of uninterpreted plans $\pi(\mathcal{A})$, where $\mathcal{A}$ is an incomplete argument in $\mathbb{P}^m \oplus \pi$
- $\mathsf{Plans}_m =$ the set of pairs $(\pi, \pi^{(\mathbb{M}^m)^+})$ with an uninterpreted plan π and its free interpretation $\pi^{(\mathbb{M}^m)^+}$
- $\mathsf{Trueplans}_m =$ the set of (correctly) interpreted plans $\pi^{\mathbb{M}^m}$ in a planning domain $\mathbb{M}^m$; that is, a set of tuples *(actions, trees, goals)* as in Definition 3.11
- $\mathsf{strict}_m =$ a set of auxiliary information: strict facts showing that some "open goals" are actually solved; or the actions supporting plan steps in pre-plans

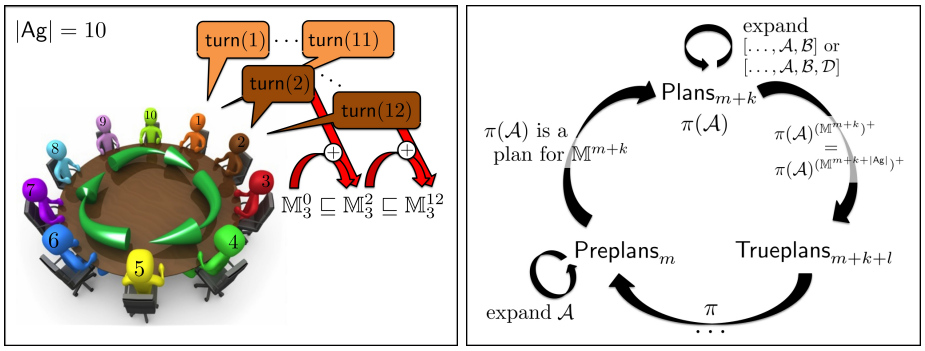

FIG. 6: (Left) A representation of the cyclic dialogues; e.g. at turn 2 agent 2 addresses to 3, who expands her initial planning domain $\mathbb{M}^0_3$ into $\mathbb{M}^2_3$; the same, at turn 12 with an expansion of $\mathbb{M}^2_3$ into $\mathbb{M}^{12}_3$. (Right) Phases in the turn-based construction of a plan: $\mathsf{Preplans}_m \to \mathsf{Plans}_{m+k} \to \mathsf{Trueplans}_{m+k+l} \to \mathsf{Preplans}_{m+k+l+1}$ and so on. Their transitions occur, resp., when a potential argument is expanded into an argument $\mathcal{A}$, so $\pi(\mathcal{A})$ appears as a plan to the agent; when after expanding potential (strict) threats, a full round occurs without further contributions; and finally, when the resulting plan can again be refined with new plan steps.

With more detail, a pre-plan contains a terminal fragment of a conceivable plan step (which might be later completed by other agents into a full plan step). The communicated argument $\mathcal{A}$ can thus be incomplete, so for the sake of evaluation, the agents will temporarily assume as strict facts some literals from $\mathsf{base}(\mathcal{A})$ to make sense of $\mathcal{A}$, i.e. as if it were an actual argument in their t-DeLP programs.

DEFINITION 4.4 (Pre-plan)
Given a planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$, and a plan π for $\mathbb{M}$, let $\mathcal{A} \subseteq \Pi \cup \Delta \cup \mathsf{post}[A]$ be arbitrary and define $A^* = \{\mathsf{e} \in A \setminus A(\pi) \mid \langle \mu_\mathsf{e}, t_\mathsf{e} \rangle \in \mathsf{base}(\mathcal{A})\}$. Then, we say that $\pi(\mathcal{A})$ is a *pre-plan for* $\mathbb{M}$ iff the following conditions hold:

(i) $A(\pi) \cup A^*$ is non-overlapping
(ii) $((\Pi \cup \mathsf{post}[A(\pi) \cup A^*]) \cup (\mathsf{base}(\mathcal{A}) \setminus \mathsf{post}[A]), \Delta)$ is a t-DeLP program
(iii) $\mathcal{A}$ is an argument in this program
(iv) $\mathsf{pre}[A^*]$ is consistent with all literals in argument steps and all solved/open goals.

We also say that $\pi([\mathcal{A}_j, \ldots, \mathcal{A}_k, \mathcal{B}, \mathcal{A}])$ is a *pre-plan* for $\mathbb{M}$ iff $[\mathcal{A}_j, \ldots, \mathcal{A}_k, \mathcal{B}]$ is a threat in $\pi^\mathbb{M}$, and the following conditions hold (for A^* defined as above):

(i) $A(\pi) \cup A^*$ is non-overlapping
(ii) $((\Pi \cup \mathsf{post}[A(\pi) \cup A^*]) \cup (\mathsf{base}(\mathcal{A}) \setminus \mathsf{post}[A]), \Delta)$ is a t-DeLP program
(iii) $\mathcal{A}$ is an argument in this program and $\sim\mathsf{concl}(\mathcal{A}) \in \mathsf{literals}(\mathcal{B})$.

Note that, for pre-plans, the tuples (*actions, trees, goals*) as defined in Table 1 for plans, are left undefined here.

Once a pre-plan $\pi(\mathcal{A})$ is completed (at turn $m + k$) into an *apparent* plan,[7] say, $\pi(\mathcal{A}^+)$ for some $\mathcal{A}^+ \supsetneq \mathcal{A}$, this is communicated as an element in the set Plans_{m+k}. At this stage, agents evaluate it by communicating free interpretations of it (again in sets of the form $\mathsf{Plans}_{m+k+1}, \ldots, \mathsf{Plans}_{m+k+l}$). The free interpretations include *pre-threats* (incomplete threats) against existing plan steps, as well as –possibly incomplete– claims against the validity of the last plan step $\mathcal{A}^+$, or of new (pre-)threats against plan steps; those claims will be called *strict pre-threats* and will be expressed with the notation for argumentation lines.

DEFINITION 4.5 (Pre-threat)
Let a plan $\pi = \pi_\varnothing(\Lambda_1, \ldots, \Lambda_n)$ for some planning domain $\mathbb{M} = (\mathbb{P}, A, G)$ be given, where $\mathbb{P} = (\Pi, \Delta)$. We say that $\mathcal{B} \subseteq \Pi \cup \Delta \cup \mathsf{post}[A(\pi)]$ is a *pre-threat* for a plan step $\Lambda_k = [\mathcal{A}, \ldots, \mathcal{A}_k]$ iff

(i) $(\Pi \cup A(\pi) \cup (\mathsf{base}(\mathcal{B}) \setminus \mathsf{post}[A]), \Delta)$ is a t-DeLP program, and
(ii) $\mathcal{B} \cup \mathsf{base}(\mathcal{B})$ is an argument in this program and $\sim\mathsf{concl}(\mathcal{B}) \in \mathsf{literals}(\mathcal{A}_k) \setminus \mathsf{base}(\mathcal{A}_k)$

This pre-threat will be denoted $[\mathcal{A}, \ldots, \mathcal{A}_k, \mathcal{B}] \in \mathcal{T}^+_{\mathbb{P} \oplus \pi}(\mathcal{A})$.

In particular, a pre-threat $[\mathcal{A}, \ldots, \mathcal{A}_k, \mathcal{B}]$ can eventually become a threat for the plan step $[\mathcal{A}, \ldots, \mathcal{A}_k]$, in which case it will be included in the usual sub-tree $\mathcal{T}^*_{\mathbb{P} \oplus \pi}(\mathcal{A})$ of plan $\pi \in \mathsf{Plans}_{(\cdot)}$.

EXAMPLE 4.6 (Lifting a table –pre-threat)
In the dialogue of Table 3 for solving Example 4.2, an incomplete pre-threat $\{\delta_6, \delta_7\}$ is suggested against the plan step $\mathcal{A}_1$ in plan π_1. This pre-threat is later completed into two threats: $\mathcal{B}_1$ in plan π_{14}, and $\mathcal{B}_2$ in plan π_{15}.

[7] Note that an agent will always send a proposal which is a plan according to herself, but it need not be so according to other agents.

In addition to the above pre-threats, we also consider claims against the fact that a proposed plan step or (pre-)threat is really an argument: after a turn, it might no longer be defeasibly minimal (i.e. if it contains a derived literal which is already a strict fact or is derivable using strict information), or might simply become inconsistent (with the strict part). If these claims, called *strict threats*, succeed, the plan or (pre-) threat is discarded.

DEFINITION 4.7 (Strict pre-threat; Strict threat)
Let $\pi = \pi_{\varnothing}(\ldots, \Lambda_k, \ldots, \Lambda_n)$ be a plan for $\mathbb{M} = (\mathbb{P}, A, G)$, for some $\mathbb{P} = (\Pi, \Delta)$, and with $\Lambda_k = [\mathcal{A}_j, \ldots, \mathcal{A}_k]$ and $\Lambda_n = [\ldots, \mathcal{A}_n]$; and let either $\mathcal{A}$ be $\mathcal{A}_n$ or a pre-threat for Λ_k, i.e. $[\mathcal{A}_j, \ldots, \mathcal{A}_k, \mathcal{A}] \in \mathcal{T}^{+}_{\mathbb{P}\oplus\pi}(\mathcal{A})$. Finally, let $\mathbb{M}' = ((\Pi', \Delta'), A', G)$ be a planning domain such that $\mathbb{M} \sqsubseteq \mathbb{M}'$.

A set $\mathcal{B} \subseteq \Pi' \cup \mathsf{post}[A(\pi)]$ is called a *strict pre-threat* for $\mathcal{A}$ in $\mathbb{M}'$ iff $\mathcal{B}$ shows that either:

(i) $\mathcal{A}$ *is not consistent*, i.e. $\sim\mathsf{concl}(\mathcal{B}) \in \mathsf{literals}(\mathcal{A}) \setminus \mathsf{base}(\mathcal{A})$

(ii) $\mathcal{A}$ *is not defeasibly minimal*, i.e. $\mathsf{concl}(\mathcal{B}) \in \mathsf{literals}(\mathcal{A}) \setminus \mathsf{base}(\mathcal{A})$, $\mathcal{B} \nsubseteq \mathcal{A}$ and $\mathcal{A}(\mathsf{concl}(\mathcal{B})) \nsubseteq \Pi \cup \mathsf{post}[A(\pi)]$

If moreover $\mathcal{B}$ is an argument in $\mathbb{P}' \oplus \pi$, then we also say that $\mathcal{B}$ is a *strict threat*. These strict (pre-)threats will be denoted as $[\ldots, \mathcal{A}, \mathcal{B}] \in \mathcal{T}^{+}_{\mathbb{P}\oplus\pi}(\cdot)$.

The free interpretation of a plan simply adds to the $\mathcal{T}^{*}$-trees in the usual interpretation (Table 1), all the pre-threats and strict pre-threats as defined above.

DEFINITION 4.8 (Free interpretation)
Let $\mathbb{M}^{m+1} = (\mathbb{P}^{m+1}, A^{m+1}, G)$ be the planning domain of agent $f(m+1)$ at turn $m+1$. Assume either that $\pi = \pi_{\varnothing}(\Lambda_1, \ldots, \Lambda_n) \in \mathsf{Preplans}_{m+1}$ is a plan for $\mathbb{M}^{m+1}$, or that $(\pi, \pi^{(\mathbb{M}^m)^+}) \in \mathsf{Plans}_m$. We define the *free interpretation of* π in $\mathbb{M}^{m+1}$ as

$$\pi^{(\mathbb{M}^{m+1})^+} = \left(A(\pi^{\mathbb{M}^{m+1}}),\ \mathsf{Trees}(\pi^{(\mathbb{M}^{m+1})^+}),\ \mathsf{goals}(\pi^{\mathbb{M}^{m+1}})\right),$$

where $\mathsf{Trees}(\pi^{(\mathbb{M}^{m+1})^+}) = \{\mathcal{T}^{+}_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k) \mid \mathcal{A}_k \text{ is an argument step }\}$, with each $\mathcal{T}^{+}_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k)$ being defined as follows:

$$\mathcal{T}^{+}_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k) = \mathcal{T}^{*}_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k) \cup \begin{cases} \text{the set of pre-threats } [\mathcal{A}_k, \ldots, \mathcal{A}_{k'}, \mathcal{B}] \text{ and their} \\ \qquad \text{strict (pre-)threats } [\mathcal{A}_k, \ldots, \mathcal{A}_{k'}, \mathcal{B}, \mathcal{D}] \\ \qquad \text{if } k \neq n \neq k' \\ \text{the set of (strict) pre-threats } {\Lambda_n}^{\frown}[\mathcal{B}] \text{ and their} \\ \qquad \text{strict (pre-)threats } {\Lambda_n}^{\frown}[\mathcal{B}, \mathcal{D}] \\ \qquad \text{if } k = n \text{ or } \Lambda_n = [\mathcal{A}_k, \ldots, \mathcal{A}_n]. \end{cases}$$

The proposed dialogues among agents, for a given multi-planner domain $\langle \mathbb{M}_a \rangle_{a\in\mathsf{Ag}}$, take place by agents communicating the contents defined in Table 2. These definitions are used in Algorithm 2 describing how each agent communicates at each turn.

In summary, each agent $f(m)$ sends in $\mathsf{turn}(m)$ lists of pre-plans, free interpretations of (presumed) plans, and verified plans to agent $f(m+1)$. The latter agent extracts the information contained in these lists and adds it to its own planning domain. Then, the

TABLE 2: A definition of the turn-based dialogue. Recall that we abbreviate agent subindices in expressions like $\mathbb{M}^{(\cdot)}_{f(m+1)}$ to actually denote $\mathbb{M}^{(\cdot)}_{a_{f(m+1)}}$; the same for Π, Δ and A. Note that $\mathbb{M}^m_a \sqsubseteq \mathbb{M}^{m+1}_a$ and $\mathbb{M}^m_a \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$ for any m and a.

$$
\begin{array}{lcl}
\mathsf{turn}(0) & = & (\ \varnothing,\ \varnothing,\ \{\pi_\varnothing\},\ \varnothing\ ) \\
\mathsf{turn}(m+1) & = & (\ \mathsf{Preplans}_{m+1},\ \mathsf{Plans}_{m+1},\ \mathsf{Trueplans}_{m+1},\ \mathsf{strict}_{m+1}\ ) \\
\hline
\mathsf{Preplans}_{m+1} & = & (\mathsf{Preplans}_m \smallsetminus \mathsf{Preplans}_{m-|\mathsf{Ag}|}) \cup \\
 & & \cup \{\pi(\mathcal{A}) \text{ is a pre-plan for } \mathbb{M}^{m+1}_{f(m+1)} \mid \pi \in \mathsf{Trueplans}_m\} \\
\mathsf{Plans}_{m+1} & = & \left\{ (\pi, \pi^{(\mathbb{M}^{m+1}_{f(m+1)})^+}) \;\middle|\; \begin{array}{c} (\pi, \pi^{(\mathbb{M}^m_{f(m)})^+}) \in \mathsf{Plans}_m, \text{ or} \\ \pi \in \mathsf{Preplans}_{m+1} \\ \text{and } \pi \text{ is a plan for } \mathbb{M}^{m+1}_{f(m+1)} \end{array} \right\} \\
\mathsf{Trueplans}_{m+1} & = & \mathsf{Trueplans}_m \cup \\
\left(\begin{array}{c}\text{Note: for } m < |\mathsf{Ag}|, \text{ let} \\ \pi^{(\mathbb{M}^{m+1-|\mathsf{Ag}|}_{f(m+1)})^+} = \varnothing.\end{array}\right) & & \cup \left\{ \pi \;\middle|\; \begin{array}{c} (\pi,\cdot) \in \mathsf{Plans}_m, \pi \text{ is a plan for } \mathbb{M}^{m+1}_{f(m+1)} \\ \text{and } \pi^{(\mathbb{M}^{m+1}_{f(m+1)})^+} = \pi^{(\mathbb{M}^{m+1-|\mathsf{Ag}|}_{f(m+1)})^+} \end{array} \right\} \\
\mathsf{strict}_{m+1} & = & \mathsf{strict}_m \ \cup \ \left(\Pi^{m+1}_{f(m+1)} \cap \bigcup\nolimits_{(\pi,\cdot)\in\mathsf{Plans}_m} \mathsf{goals}(\pi^{\mathbb{M}^m})\right) \cup \\
 & & \cup \left\{ \mathsf{e} \in A^{m+1} \;\middle|\; \begin{array}{c} \langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle \in \mathsf{base}(\mathcal{A}), \text{ for} \\ \text{some } \pi(\mathcal{A}) \in \mathsf{Preplans}_{m+1} \end{array} \right\} \\
\hline
\mathbb{M}^{m+1}_a & = & \begin{cases} \mathbb{M}^m_a & \text{if } a \neq f(m+2) \\ (\mathbb{P}^{m+1}_{f(m+2)}, A^{m+1}_{f(m+2)}, G) & \text{if } a = f(m+2) \end{cases} \\
 & & \text{where} \\
A^{m+1}_{f(m+2)} & = & A^m_{f(m+2)} \cup \{\mathsf{e} \in \mathsf{strict}_{m+1} \mid \mathsf{e} \text{ is an action}\} \\
\Pi^{m+1}_{f(m+2)} \cup \Delta^{m+1}_{f(m+2)} & = & \Pi^m_{f(m+2)} \ \cup \ \Delta^m_{f(m+2)} \ \cup \ \{\langle \ell, t\rangle \mid \langle \ell, t\rangle \in \mathsf{strict}_{m+1}\} \cup \\
\left(\begin{array}{c}\text{Note: if } \delta \text{ has } \leftarrow, \\ \delta \in \Pi^{m+1}_{f(m+2)}; \\ \text{and if } \delta \text{ has } \prec, \\ \delta \in \Delta^{m+1}_{f(m+2)}\end{array}\right) & & \left\{ \delta \in \begin{array}{c} \Pi^{m+1}_{f(m+1)} \\ \cup \\ \Delta^{m+1}_{f(m+1)} \end{array} \;\middle|\; \begin{array}{l} \delta \in \mathcal{A} \smallsetminus \mathsf{base}(\mathcal{A}) \\ \text{for some } \pi(\mathcal{A}) \in \mathsf{Preplans}_{m+1}, \text{or} \\ \Lambda^{\cap}[\mathcal{A}], \Lambda^{\cap}[\mathcal{A},\cdot] \in \mathcal{T}^+_{\mathbb{P}^{m+1}_{f(m+1)} \oplus \pi}(\cdots) \\ \text{with } (\pi,\cdot) \in \mathsf{Plans}_{m+1} \end{array} \right\}
\end{array}
$$

process of generating and communicating pre-plans, free interpreations and verified plans is repeated using the new information available. This definition in Table 2 of the contents communicated in $\mathsf{turn}(m)$, and the resulting update into $\mathbb{M}^m_{f(m+1)}$ is used by the next algorithm describing a given agent's contribution to the dialogue.

Algorithm 2: Dialogue for agent a in t-DeLP planning with $\mathsf{Ag} = \{1, \ldots, n\}$.

Data: $\mathbb{M}^0_a$;
Result: π; or fail;
initialization: $m = 0$ and flag = false and $\mathsf{turn}(0) = (\varnothing, \varnothing, \{\pi_\varnothing\}, \varnothing)$
and $\mathsf{turn}(a - |\mathsf{Ag}|) = \varnothing$ and, for $a = 1$, $\mathbb{M}^1_1 = \mathbb{M}^0_1$;
while $\mathsf{turn}(m) \neq \mathsf{turn}(m - |\mathsf{Ag}|)$ *and* $\mathsf{flag} = \mathit{false}$ **do**
  while $f(m+1) \neq a$ **do**
    set $\mathbb{M}^{m+1}_a = \mathbb{M}^m_a$;
    set $m = m + 1$;
  end
  wait for message $\mathsf{turn}(m)$ from agent $f(a-1)$;
  set $\mathsf{ToTest} = \mathsf{Trueplans}_m$;
  while $\mathsf{ToTest} \neq \varnothing$ *and* $\mathsf{flag} = \mathit{false}$ **do**
    select π from ToTest;
    if $\mathsf{goals}(\pi^{\mathbb{M}^m_a}) = \varnothing$ *and* $\mathsf{threats}(\pi^{\mathbb{M}^m_a}) = \varnothing$ **then**
      set $\mathsf{flag} = \mathit{true}$
    else
      delete π from ToTest
    end
  end
  if $\mathsf{flag} = \mathit{false}$ **then**
    set $\pi = \mathsf{undefined}$
  end
  compute $\mathbb{M}^{m+1}_a$;
  compute $\mathsf{turn}(m+1)$;
  send $\mathsf{turn}(m+1)$ to agent $f(a+1)$;
end

See Table 3 for an illustration of the dialogue taking place between the agents a_1 and a_2 when solving this problem using Algorithm 2.

4.3 Soundness and Completeness of the Dialogue-based algorithm

Before proceeding with the results for soundness and completeness of the dialogue-based algorithm, note that planning domains $\mathbb{M}^n_a$ keep expanding during the dialogues, but always bounded by the centralised planning domain, i.e. $\mathbb{M}^n_a \sqsubseteq \mathbb{M}^{n+1}_a \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. The reader is referred to the Appendix, for proofs of the results in this section.

LEMMA 4.9
Let $\pi = \pi_n$ be a plan for two planning domains $\mathbb{M}, \mathbb{M}'$, both of the form $((\cdot,\cdot),\cdot, G)$ and with $\mathbb{M} \sqsubseteq \mathbb{M}'$, and moreover satisfying $\pi^{\mathbb{M}}_k = \pi^{\mathbb{M}'}_k$, for each $k \leq n$. Then, if $\pi(\mathcal{A})$ is again a plan for both $\mathbb{M}, \mathbb{M}'$, we have

$$A(\pi(\mathcal{A})^{\mathbb{M}}) = A(\pi(\mathcal{A})^{\mathbb{M}'}) \qquad \text{and} \qquad \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}'}) = \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}}) \setminus \Pi'.$$

TABLE 3: Dialogue for a decentralised version of Example 4.2. The notation used is, e.g., as follows: π_{mnk} denotes a plan $\pi_{\varnothing}(\mathcal{A}_m, \mathcal{A}_n, \mathcal{A}_k)$; and $\mathbb{M}^n_k, \mathbb{P}^n_k$ denote $\mathbb{M}^n_{a_k}, \mathbb{P}^n_{a_k}$.

turn	informal dialogue (for Example 4.2)	formal dialogue
0, –	The empty plan $\pi_{\varnothing}$ is available.	$\pi_{\varnothing} \in \mathsf{Trueplans}_0$
1, a_1	Let us say the vase does not break.	$\pi_1 \in \mathsf{Preplans}_1$
	But if the table is non-horizontal, and the vase is on it, the vase will break! (a_2 now learns δ_7)	$(\pi_1, \pi_1^{(\mathbb{M}_1^1)^+}) \in \mathsf{Plans}_1$ with $[\mathcal{A}_1, \{\delta_6, \delta_7\}] \in \mathcal{T}^+_{\mathbb{P}_1^1 \oplus \pi_1}(\mathcal{A}_1)$
	I might lift.N at $t = 9$.	$\pi_4 \in \mathsf{Preplans}_1$
2, a_2	I might lift.S at $t = 9$.	$\pi_5 \in \mathsf{Preplans}_2$
3, a_1	We agree that π_1, π_4 are plans.	$\pi_1, \pi_4 \in \mathsf{Trueplans}_3$
4, a_2	We agree that π_5 is a plan.	$\pi_5 \in \mathsf{Trueplans}_4$
	You might lift.N at $t = 9$ on π_1.	$\pi_{14} \in \mathsf{Preplans}_4$
	Or I might lift.S at $t = 9$ on π_1 or π_4.	$\pi_{15}, \pi_{45} \in \mathsf{Preplans}_4$
	In π_{14} the vase will break!	$[\mathcal{A}_1, \mathcal{B}_1] \in \mathcal{T}^+_{\mathbb{P}_2^4 \oplus \pi_{14}}(\mathcal{A}_1)$
	In π_{15} the vase will break!	$[\mathcal{A}_1, \mathcal{B}_2] \in \mathcal{T}^+_{\mathbb{P}_2^4 \oplus \pi_{15}}(\mathcal{A}_1)$
5, a_1	I might lift.N at $t = 9$ on π_5.	$\pi_{54} \in \mathsf{Preplans}_5$
6, a_2	We agree that π_{15}, π_{45} are plans.	$\pi_{15}, \pi_{45} \in \mathsf{Trueplans}_6$
7, a_1	You might lift.N at $t = 9$ on π_{15}	$\pi_{154} \in \mathsf{Preplans}_7$
	In π_{154}, the vase will break!	$[\mathcal{A}_1, \mathcal{B}_1], \ldots \in \mathcal{T}^+_{\mathbb{P}_1^7 \oplus \pi_{154}}(\mathcal{A}_1)$
	We agree that π_{54} is a plan.	$\pi_{54} \in \mathsf{Trueplans}_7$
⋮	⋮	⋮
9, a_1	We agree that π_{154} is a plan.	$\pi_{154} \in \mathsf{Trueplans}_9$
10, a_2	$[\mathcal{A}_1, \mathcal{B}_1, \mathcal{A}_3]$ solves threat in π_{154}.	$\pi_{1543} \in \mathsf{Preplans}_{10}$
	$\{\delta_r\}$ ($r \in \{1, 2\}$) is a pre-threat for $\mathcal{A}_3$.	$[\mathcal{A}_1, \mathcal{B}_1, \mathcal{A}_3, \{\delta_r\}]$,
	$[\mathcal{A}_1, \mathcal{B}_2]$ in π_{1543} is not yet solved.	$[\mathcal{A}_1, \mathcal{B}_2] \in \mathcal{T}^+_{\mathbb{P}_2^{10} \oplus \pi_{1543}}(\mathcal{A}_1)$
⋮	⋮	⋮
12, a_2	We agree that π_{1543} is a plan.	$\pi_{1543} \in \mathsf{Trueplans}_{12}$
13, a_1	$[\mathcal{A}_1, \mathcal{B}_2, \mathcal{A}_3]$ solves threat in π_{1543}.	$\pi_{15433} \in \mathsf{Preplans}_{13}$
	$\{\delta_r\}$ ($r \in \{1, 2\}$) is a pre-threat for $\mathcal{A}_3$.	$[\ldots, \mathcal{A}_3, \{\delta_r\}] \in \mathcal{T}^+_{\mathbb{P}_1^{13} \oplus \pi_{15433}}(\mathcal{A}_1)$
⋮	⋮	⋮
15, a_1	We see that π_{15433} is a plan.	$\pi_{15433} \in \mathsf{Trueplans}_{15}$
	(And I think π_{15433} is a solution.)	Terminating Cond. in $\mathbb{M}_1^{15}$.
16, a_2	(I think that π_{15433} is a solution.)	Terminating Cond. in $\mathbb{M}_2^{16}$.

THEOREM 4.10 (Soundness)
Let π be the output of the dialogue-based plan search algorithm for some given multi-planner domain $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$. Then $A(\pi)$ is a solution for $\mathbb{M}_{\mathsf{Ag}}$.

Moreover, it can also be seen that the output is a solution for the resulting planning domain of each agent $a \in \mathsf{Ag}$.

COROLLARY 4.11
Let $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$ be a multi-planner domain, and let π_n be the output of the dialogue-based algorithm for $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$. Assume that $\pi_n \in \mathsf{Trueplans}_{m_n}$ with m_n minimal with this property. Then:

- $\pi_n \in \mathsf{Trueplans}_{m_n} \cap \ldots \cap \mathsf{Trueplans}_{m_n+|\mathsf{Ag}|-1}$, and
- $A(\pi_n)$ is a solution for any $\mathbb{M}^{m_n+j}_{f(m_n+j)}$ with $0 \leq j < |\mathsf{Ag}|$.

In addition, in this study of the dialogue-based plan search algorithm one can also show that the algorithm is complete.

THEOREM 4.12 (Completeness)
Let $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$ be a multi-planner domain. If a solution A' exists for the centralised domain $\mathbb{M}_{\mathsf{Ag}}$, then the dialogue-based algorithm terminates with an output π.

Finally, assume that no solution exists for a particular $\mathbb{M}_{\mathsf{Ag}}$. Since the goals are bounded, there are only finitely-many rules, facts and actions, and so the set of plan steps and possible threats is also finite. Hence, both algorithms for centralised and cooperative planning will terminate with a fail message after exploring all their combinations.

COROLLARY 4.13 (Termination)
Algorithms 1 and 2 terminate given any input, a planning domain $\mathbb{M}$ and a multi-planner domain $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$, respectively.

5 Conclusions and Related Work

In this paper we propose a type of dialogue-based algorithm for distributed plan search in cooperative planning problems expressible in a t-DeLP-based planning system. The main motivation for a dialogue-based approach to cooperative planning among autonomous agents (vs. a centralised approach) is the advantage of preserving the privacy of their information beyond the needs for the common goals. The underlying logic-based planning system combines a traditional update function for temporal planning with an (argumentative) temporal defeasible logical system called t-DeLP. This logical system adds non-monotonic temporal reasoning to actions, thus allowing for complex indirect effects. Our contribution is mainly theoretical and consists of showing that the dialogue-based plan search methods are sound and complete for t-DeLP planning.

The results of the present paper are similar to those from [19, 14, 13], but replacing DeLP [9] by t-DeLP [15], and also the partial order planner of [8] with the temporal linear planner from [17]. While POP planning systems are more flexible, the underlying logic DeLP is less expressive given the implicit time approach. The present paper can also be related to other works combining a multi-agent approach,

argumentative reasoning and planning methods for decision-making. None of these works, though, exhibits all of these features. For example, abstract argumentation [6] has been used to reason about conflicting plans and generate consistent sets of goals [1, 11, 21]. None of these works apply to a multi-agent environment. A proposal for dialogue-based centralised planning is that of [24], but no argumentation is made use of. The work in [3] presents a dialogue based on an argumentation process to reach agreements on plan proposals. Unlike our focus on an argumentative and stepwise construction of a plan, this latter work is aimed at handling the interdependencies between agents' plans. In a similar vein, [12] presents an argumentation scheme to propose and justify plans using critical questions.

In the area of traditional automated planning, several extensions of classical planning exist in either of the directions taken in the present paper. For example, the literature on temporal planning is quite rich (see e.g. [10, Ch. 14]), though mainly based on simple, monotonic reasoning. Multi-agent planning for collaborative scenarios has also been studied in a dialogue-based form [25], again built on classical or temporal planners. For multi-agent planning, on the other hand, most existing works differ from the present approach by permitting the concurrent distribution of plan search –in contrast to the unique, sequential dialogues considered here. The latter aspect prevents us from taking advantage of faster methods based on concurrent planning, where different plans proposals take place, but grants soundness and completentess when no assumptions are made upon the distribution of knowledge among agents. In this sense, additional assumptions might suffice to reconcile argumentation-based planning with a concurrent approach to multi-agent planning: one might assume, for example, that the information (indirect effects, threats) relevant to a combination of actions is distributed among the agents responsible for those actions. Under this assumption, a concurrent version of the proposed dialogue protocol might preserve the soundness and completeness of the current algorithm. This is left for future work, together with computational and more practical aspects like the efficiency, applicability and scalability of the approach.

6 Appendix: Proofs.

Proof of Theorem 3.14 (Soundness of BFS plan search).

PROOF. Let $\pi = \pi_n = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be the output of the BFS algorithm. We prove that $A(\pi)$ is a solution, i.e. $G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A(\pi))$, by showing the stronger claim

$$\bigcup_{0 \le k \le n} \mathsf{goals}(\pi_k) \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A(\pi)) \qquad (\text{note that } \; G = \mathsf{goals}(\pi_0))$$

By Lemma 3.5, we only need to check that, for any literal $\langle \ell, t \rangle$ and plan π_k

$$(\star 1) \quad \text{if } \; \langle \ell, t \rangle \in \mathsf{goals}(\pi_k), \quad \text{then } \; \langle \ell, t \rangle \in \mathsf{warr}((\Pi, \Delta) \diamond A(t))$$

where $A(t) = \{\, \mathsf{e} \in A(\pi) \mid t_\mathsf{e} \le t \,\}$. The proof of $(\star 1)$ is by induction on t, together with the auxiliary claim that actions are executable:

$$(\star 2) \quad (\Pi \oplus A(t), \Delta) = (\Pi, \Delta) \diamond A(t)$$

(Base Case $t = 0$) The proof of ($\star 2$) is straightforward provided that $A(0) = \varnothing$. The proof of ($\star 1$) is similar to the inductive case and will not be repeated.

(Inductive Case $t \Rightarrow t+1$) Assume (Inductive Hypothesis) the claims ($\star 1$) and ($\star 2$) for each $t' \leq t$. Let us show first ($\star 2$) for $t+1$. Let E_{t+1} be the set of actions in $A(\pi)$ with an effect of the form $\langle \cdot, t+1 \rangle$, so we have $A(t+1) = A(t) \cup E_{t+1}$. Then,

$$\begin{array}{lll} (\Pi \oplus A(t+1), \Delta) & = & ((\Pi \oplus A(t)) \oplus E_{t+1}, \Delta) \\ = \quad ((\Pi \oplus A(t)), \Delta) \diamond E_{t+1} & & \text{(since } \mathsf{pre}[E_{t+1}] \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A(t))) \\ = \quad ((\Pi, \Delta) \diamond A(t)) \diamond E_{t+1} & = & (\Pi, \Delta) \diamond (A(t) \cup E_{t+1}) \quad = \quad (\Pi, \Delta) \diamond A(t+1) \end{array}$$

For the ($\star 1$) claim, let $\langle \ell, t+1 \rangle$ be an arbitrary goal in $\bigcup_{1 \leq k \leq n} \mathsf{goals}(\pi_k)$. Since $\mathsf{goals}(\pi_n) = \varnothing$, let $k, k' \leq n$ be minimal with $\langle \ell, t+1 \rangle \in \mathsf{goals}(\pi_k)$ and resp. $\langle \ell, t+1 \rangle \notin \mathsf{goals}(\pi_{k'})$. By construction, $\mathcal{A}_{k'}$ can only be an argumentation step and moreover, such that either $\langle \ell, t+1 \rangle = \mathsf{concl}(\mathcal{A}_{k'})$ or $\langle \ell, t+1 \rangle \in \Pi_{\mathsf{f}}$ or finally $\langle \ell, t+1 \rangle \in \mathsf{OldGoals}(\pi_{k'-1})$. The latter is impossible by the definition of k'. And so is $\langle \ell, t+1 \rangle \in \Pi_{\mathsf{f}}$, since $\langle \ell, t+1 \rangle \in \mathsf{goals}(\pi_k)$. So it remains to check the ($\star 1$) claim for the former possibility. That is, it remains to prove that

$$\mathcal{A}_{k'} \text{ is an argument for } \langle \ell, t+1 \rangle \quad \Rightarrow \quad \langle \ell, t+1 \rangle \in \mathsf{warr}(\mathbb{P} \diamond A(t+1))$$

This can be shown by combining the assumption $\mathsf{threats}(\pi_n) = \varnothing$ and the construction of $\mathcal{T}^*_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$, where each interfering argument has a defeater, extracted from an existing defeater in $\mathcal{T}_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$. This implies that $\mathcal{A}'_k$ is undefeated in $\mathcal{T}^*_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$.

Finally, it can also be easily shown by the construction of the $\mathcal{T}^*$ trees that the undefeated status for an argument is equivalent between $\mathcal{T}^*$ and the full dialectical tree $\mathcal{T}$, so we conclude the proof as follows

$$\begin{array}{ll} \langle \ell, t+1 \rangle \in \mathsf{warr}(\mathbb{P} \oplus \pi_n) & \text{(def. of warrant)} \\ \langle \ell, t+1 \rangle \in \mathsf{warr}(\mathbb{P} \oplus A(t+1)) & \text{(by Lemma 3.5)} \\ \langle \ell, t+1 \rangle \in \mathsf{warr}(\mathbb{P} \diamond A(t+1)) & \text{(by the above claim } (\star 2) \text{ for } t+1) \end{array}$$

This concludes the proof of the Inductive Case. As we mentioned the proof of this Theorem can be completed as follows:

$$\begin{array}{rcll} \langle \ell, t \rangle \in G & \Rightarrow & \langle \ell, t \rangle \in \mathsf{warr}(\mathbb{P} \diamond A(t)) & \text{(the above inductive proof)} \\ & \Rightarrow & \langle \ell, t \rangle \in \mathsf{warr}(\mathbb{P} \diamond \pi_n) & \text{(by Lemma 3.5)} \\ G & \subseteq & \mathsf{warr}(\mathbb{P} \diamond \pi_n) & \text{(since } \langle \ell, t \rangle \text{ is arbitrary)} \end{array}$$

∎

Proof of Theorem 3.15 (Completeness of BFS plan search).

Proof. Let $A' \subseteq A$ be a solution for a given planning domain $\mathbb{M} = (\mathbb{P}, A, G)$, so $G \subseteq \mathsf{warr}(\mathbb{P} \diamond A')$. Without loss of generality, we assume that this set A' is a $\subseteq$-minimal solution. We proceed to define by induction a plan of the form $\pi_n = \pi_\varnothing(\Lambda_1, \ldots, \Lambda_n)$ that satisfies the Terminating Condition and whose set of actions is $A(\pi_n) = A'$.

In order to generate π_n, we first define $G^+ = G \cup \mathsf{pre}[A']$. Notice that $\mathsf{pre}[A'] \subseteq \mathsf{warr}(\mathbb{P} \diamond A')$, because A' is a $\subseteq$-minimal solution. Thus, for each $\langle \ell, t \rangle \in G^+$, there exists an argument $\mathcal{A}_{\langle \ell, t \rangle}$ for $\langle \ell, t \rangle$ undefeated in $\mathcal{T}_{\mathbb{P} \oplus A'}(\mathcal{A}_{\langle \ell, t \rangle})$. Define $\mathsf{ArgSteps} =$

$\{\mathcal{A}_{\langle \ell,t\rangle} \mid \langle \ell, t\rangle \in G^+\}$. Another consequence of the $\subseteq$-minimality of A' is that $\mathbb{P} \diamond A' = \mathbb{P} \oplus A'$, which can be seen by induction on the sets $A(t)$ of actions executed before or at t. In order to obtain the remaining plan steps in the desired plan, define by simultaneous induction, a pair of minimal sets Steps and $\mathsf{Threats}$ as containing:

$$
\begin{array}{rcll}
\{[\mathcal{A}_{\langle \ell,t\rangle}]\}_{\mathcal{A}_{\langle \ell,t\rangle} \in \mathsf{ArgSteps}} & \subseteq & \mathsf{Steps} & \\
[\mathcal{A}_{\langle \ell,t\rangle}, \ldots, \mathcal{A}, \mathcal{B}] & \in & \mathsf{Threats} & \text{for any pair } [\mathcal{A}_{\langle \ell,t\rangle}, \ldots, \mathcal{A}] \in \mathsf{Steps} \\
 & & & \text{and } [\mathcal{A}_{\langle \ell,t\rangle}, \ldots, \mathcal{A}, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}\oplus A'}(\mathcal{A}_{\langle \ell,t\rangle}) \\
[\mathcal{A}_{\langle \ell,t\rangle}, \ldots, \mathcal{B}, \mathcal{C}] & \in & \mathsf{Steps} & \text{for any } [\mathcal{A}_{\langle \ell,t\rangle}, \ldots, \mathcal{B}] \in \mathsf{Threats} \text{ and a} \\
 & & & \text{unique } [\mathcal{A}_{\langle \ell,t\rangle}, \ldots, \mathcal{B}, \mathcal{C}] \in \mathcal{T}_{\mathbb{P}\oplus A'}(\mathcal{A}_{\langle \ell,t\rangle}) \\
 & & & \text{such that } \mathcal{C} \text{ is undefeated in this line}
\end{array}
$$

Now, we define $\mathcal{T}^*(\mathcal{A}_{\langle \ell,t\rangle}) = \{\Lambda \in \mathsf{Steps} \cup \mathsf{Threats} \mid \Lambda = [\mathcal{A}_{\langle \ell,t\rangle}, \ldots]\}$. The above construction should make it clear that $\mathcal{T}^*(\mathcal{A}_{\langle \ell,t\rangle}) \sqsubseteq \mathcal{T}_{\mathbb{P}\diamond A'}(\mathcal{A}_{\langle \ell,t\rangle})$. Moreover, the above fact $\mathbb{P} \diamond A' = \mathbb{P} \oplus A'$, implies that $\mathcal{A}_{\langle \ell,t\rangle}$ is an argument in $\mathbb{P} \oplus A'$, so $\mathcal{T}_{\mathbb{P}\oplus A'}(\mathcal{A}_{\langle \ell,t\rangle})$ is defined. Moreover, $\mathbb{P} \diamond A' = \mathbb{P} \oplus A'$ and the previous fact $\mathcal{T}^*(\mathcal{A}_{\langle \ell,t\rangle}) \sqsubseteq \mathcal{T}_{\mathbb{P}\diamond A'}(\mathcal{A}_{\langle \ell,t\rangle})$ jointly imply that $\mathcal{T}^*(\mathcal{A}_{\langle \ell,t\rangle}) \sqsubseteq \mathcal{T}_{\mathbb{P}\oplus A'}(\mathcal{A}_{\langle \ell,t\rangle})$ as well.

We proceed with the definition of the plan by induction on the plan construction: (Base Case π_0). Clearly, $\pi_0 = \pi_\varnothing$ is a plan for $\mathbb{M}$.

(Inductive Case $m \Rightarrow m+1$). Let $\pi_m = \pi_\varnothing(\Lambda_1, \ldots, \Lambda_m)$ be a plan for $\mathbb{M}$ with $\Lambda_1, \ldots, \Lambda_m \in \mathsf{Steps}$. The proof is by cases.

(Sub-case $\mathsf{threats}(\pi_m) \neq \varnothing$) Since A' is a solution, for each existing threat Λ there will exist least a defeater $\lambda^{\cap}[\mathcal{C}]$, supported from actions in A'. It is routine to check that conditions (i)-(iv) from the definition of threat resolution move apply to $\pi_m(\mathcal{C})$.

(Sub-case $\mathsf{threats}(\pi_m) = \varnothing$ and $\mathsf{goals}(\pi_m) \neq \varnothing$) Let $\langle \ell, t\rangle \in \mathsf{goals}(\pi_m)$. In case $\langle \ell, t\rangle \in G$, some argument step $[\mathcal{A}_{\langle \ell,t\rangle}] \in \mathsf{ArgSteps} \subseteq \mathsf{Steps}$ exists for $\langle \ell, t\rangle$. On the other hand, if $\langle \ell, t\rangle \notin G$, then by the definition of $\mathsf{goals}(\cdot)$ in Def. 3.11, we must have $\langle \ell, t\rangle \in \mathsf{pre}(\mathsf{e})$ for some $\mathsf{e} \in A(\pi_m) \subseteq A'$, in which case by definition we have $[\mathcal{A}_{\langle \ell,t\rangle}] \in \mathsf{Steps}$. Using the Sub-Arguments and Direct Consistency postulates, it can be routinely checked that conditions (i)-(iv) of argument steps hold for $\pi_m(\mathcal{A}_{\langle \ell,t\rangle})$.

(Sub-case $\mathsf{threats}(\pi_m) = \varnothing$ and $\mathsf{goals}(\pi_m) = \varnothing$) In this case, we want to show that $A(\pi_m) = A'$. We have that π_m is a plan for $\mathbb{M}$ (by the Inductive Hypothesis) satisfying the Terminating Condition (by the Sub-Case assumption). Thus, we can apply the Soundness Theorem 3.14, and conclude that π_m is a solution for $\mathbb{M}$. Clearly, by construction of $A(\pi_m)$ from Steps (i.e. from A'), we have that $A(\pi_m) \subseteq A'$. This, together with the facts that $A(\pi_m)$ is a solution and that A' is a $\subseteq$-minimal solution implies that $A(\pi_m) = A'$.

∎

Proof of Lemma 4.9.

Proof. For the identity on actions, a minimal set of actions supporting $\mathcal{A}$ in $\mathbb{M}$ will exist and be minimal in $\mathbb{M}'$ as well, since $\mathbb{M} \sqsubseteq \mathbb{M}'$ and Π' does not contain effects $\langle \mu_{\mathsf{e}}, \cdot\rangle$. For the claim on goals, we show the case where $\mathcal{A}$ is an argument step (for threat resolution moves, the proof is analogous).

$$
\begin{aligned}
& \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}'}) \\
= \;& (\mathsf{goals}(\pi^{\mathbb{M}'}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi' \cup \mathsf{OldGoals}(\pi^{\mathbb{M}'}))) && \text{(by Def. 3.11)} \\
= \;& (\mathsf{goals}(\pi^{\mathbb{M}}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi' \cup \mathsf{OldGoals}(\pi^{\mathbb{M}'}))) \\
& && \text{(since } \pi^{\mathbb{M}'} = \pi^{\mathbb{M}} \text{ implies } \mathsf{goals}(\pi^{\mathbb{M}'}) = \mathsf{goals}(\pi^{\mathbb{M}})) \\
= \;& (\mathsf{goals}(\pi^{\mathbb{M}}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi' \cup \mathsf{OldGoals}(\pi^{\mathbb{M}}))) \\
& \text{(since } {\pi_0}^{\mathbb{M}'} = {\pi_0}^{\mathbb{M}}, \ldots, {\pi_n}^{\mathbb{M}'} = {\pi_n}^{\mathbb{M}} \text{ implies } \mathsf{OldGoals}(\pi^{\mathbb{M}'}) = \mathsf{OldGoals}(\pi^{\mathbb{M}})) \\
= \;& (\mathsf{goals}(\pi^{\mathbb{M}}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi \cup \Pi' \cup \mathsf{OldGoals}(\pi^{\mathbb{M}}))) \\
& && \text{(since } \Pi \subseteq \Pi' \text{ implies } \Pi \cup \Pi' = \Pi') \\
= \;& ((\mathsf{goals}(\pi^{\mathbb{M}}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi \cup \mathsf{OldGoals}(\pi^{\mathbb{M}}))) \setminus \Pi' \\
= \;& \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}}) \setminus \Pi' && \text{(by Def. 3.11)}
\end{aligned}
$$

∎

Proof of Theorem 4.10 (Soundness of the dialogue-based planning algorithm).

Proof. Let π_n denote the output sequence $\pi_n = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ for an arbitrary planning domain $\mathbb{M}_a^{m_n}$; and as usual, let π_k denote its initial fragment $\pi_k = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_k)$. By definition of $\mathsf{Trueplans}_{m_n}$, a sequence of turns $m_0 < \ldots < m_n$ must exist satisfying $\pi_k \in \mathsf{Trueplans}_{m_k}$, and with each m_k minimal with this property. We can assume, without loss of generality, that $\mathbb{M}^{m_k}$ is the planning domain of an initially fixed agent a –just take the next domain $\mathbb{M}^m$ with $m_k \geq m$ and $f(m) = a$. So, from here on, we simply omit subindexes.

The proof is by induction on the length k of the plans π_k. We show that the next two claims hold for each planning domain $\mathbb{M}$ satisfying $\mathbb{M}^{m_k} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$:

(1) π_k is a plan for $\mathbb{M}$ (2) ${\pi_k}^{\mathbb{M}^{m_k}} = {\pi_k}^{\mathbb{M}} = {\pi_k}^{\mathbb{M}_{\mathsf{Ag}}}$

This suffices to prove that $\pi_0, \ldots, \pi_n$ are plans for $\mathbb{M}_{\mathsf{Ag}}$, and that no goals and threats remain in π_n for either $\mathbb{M}^{m_n}$ or $\mathbb{M}_{\mathsf{Ag}}$. Finally, using Theorem 3.14, one can obtain that $A(\pi_n)$ is a solution for $\mathbb{M}_{\mathsf{Ag}}$.

(Base Case $k = 0$) The two claims (1)-(2) are obvious for $\pi_0 = (\varnothing, \varnothing, G)$ and $m_0 = 0$, since this plan is identically interpreted among any planning domain $\mathbb{M}$ of the form $((\cdot, \cdot), \cdot, G)$.

(Inductive Case $k \Rightarrow k + 1$) Assume (Inductive Hypothesis) that (1) and (2) hold for $\pi_0, \ldots, \pi_k$, and any planning domain $\mathbb{M}'$ with $\mathbb{M}^{m_k} \sqsubseteq \mathbb{M}' \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. We show that (1) and (2) hold for the plan $\pi_{k+1} \in \mathsf{Trueplans}_{m_{k+1}}$ and any $\mathbb{M}$ satisfying $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. We only show the case of argument step $\pi_{k+1} = \pi_k(\mathcal{A}_{k+1})$, since the remaining case can be similarly proved.

Claim (1). First, π_k is a plan for $\mathbb{M}^{m_{k+1}}$ by (2) and $\mathbb{M}^{m_k} \sqsubseteq \mathbb{M}^{m_{k+1}}$. Second, $\mathsf{concl}(\mathcal{A}_{k+1}) \in \mathsf{goals}(\pi_k^{\mathbb{M}^{m_{k+1}}})$ since this set is identical to $\mathsf{goals}(\pi_k^{\mathbb{M}^{m_k}})$ and contains $\mathsf{concl}(\mathcal{A}_{k+1})$ –since otherwise π_{k+1} would not have been generated. Third, $\mathcal{A}_{k+1}$ is an argument in $\mathbb{P}^{m_{k+1}}$ as can easily seen by induction, from the conclusion to the base/actions: the sets $\mathsf{Preplans}_{m'}$ with $m_k < m' < m_{k+1}$ show $\mathcal{A}_{k+1}$ is contained in $\Pi^{m_{k+1}} \cup \Delta^{m_{k+1}}$, while if $\mathcal{A}_{k+1}$ was not an argument in $\mathbb{P}^{m_{k+1}} \oplus A(\pi_{k+1})$, then π_{k+1} would not be in $\mathsf{Trueplans}_{m_{k+1}}$. All these results for $\mathbb{M}^{m_{k+1}}$ are preserved to

arbitrary planning domains $\mathbb{M}$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$.

Claim (2). The identity ${\pi_{k+1}}^{\mathbb{M}^{m_{k+1}}} = {\pi_{k+1}}^{\mathbb{M}} = {\pi_{k+1}}^{\mathbb{M}_{\mathsf{Ag}}}$ for an arbitrary planning domain $\mathbb{M}$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$ is shown at the level of their components: actions, sub-trees and open goals.

For *actions*, the Inductive Hypothesis for (2) implies $A({\pi_k}^{\mathbb{M}^k}) = A({\pi_k}^{\mathbb{M}^{k+1}}) = A({\pi_k}^{\mathbb{M}}) = A({\pi_k}^{\mathbb{M}_{\mathsf{Ag}}})$. Moreover, by the above claim (1) in the inductive case, π_{k+1} is a plan for arbitrary $\mathbb{M}$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. The latter two facts jointly permit to apply Lemma 4.9 and conclude that $A(\pi_{k+1}^{\mathbb{M}})$ is identical for any $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$.

For *goals*, Lemma 4.9 implies that $\mathsf{goals}({\pi_{k+1}}^{\mathbb{M}}) = \mathsf{goals}({\pi_{k+1}}^{\mathbb{M}^{m_{k+1}}}) \smallsetminus \Pi$; so, if these sets are not the same, the former contains some $\langle \ell, t \rangle \in \Pi \smallsetminus \Pi^{m_{k+1}}$. Using $\Pi \subseteq \Pi_{\mathsf{Ag}}$, we can reach a contradiction: $\langle \ell, t \rangle \in \mathsf{strict}_{m_{k+1}-1} \subseteq \Pi^{m_{k+1}}$.

For *sub-trees* $\mathcal{T}^{*}_{\mathbb{P} \oplus \pi_{k+1}}(\mathcal{A})$, note first that the set of arguments $\mathcal{A}$ for which these sub-trees are defined are the same among those planning domains $\mathbb{M} = (\mathbb{P}, A, G)$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. (Since claims to the contrary -strict threats- for some $\mathcal{A}$ would have been found before m_{k+1}.) Moreover, each of these trees are identical among these planning domains. The reason is that, otherwise, given the addition of a unique threat resolution move for each threat, there would be a threat $\Lambda^{\frown}[\mathcal{B}]$ in the tree in some $\mathbb{M}$ but not in some other $\mathbb{M}'$. But this is impossible, since Λ is a plan step in π_{k+1} according to both $\mathbb{M}$ and $\mathbb{M}'$, and hence $\mathcal{B}$ will have been built before the turn m_{k+1} where $\pi_{k+1} \in \mathsf{Trueplans}_{m_{k+1}}$. For the same reasons as above (for $\mathcal{A}$) this $\mathcal{B}$ must be a threat to π_{k+1} according to both planning domains $\mathbb{M}, \mathbb{M}'$ provided $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M}, \mathbb{M}'$.

Let us just note that for the case where $\mathcal{A}_{k+1}$ is a threat resolution move, i.e. a plan step of the form $[\mathcal{A}_i, \ldots, \mathcal{B}, \mathcal{A}_{k+1}]$, the major change is to use the Inductive Hypothesis for the identity $\mathsf{Trees}({\pi_k}^{\mathbb{M}^{m_{k+1}}}) = \mathsf{Trees}({\pi_k}^{\mathbb{M}})$, in order to prove that $[\mathcal{A}_i, \ldots, \mathcal{B}]$ exists in $\pi_k^{\mathbb{M}}$ (for arbitrary $\mathbb{M}$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$). The proof that $[\mathcal{A}_i, \ldots, \mathcal{B}, \mathcal{A}_{k+1}]$ is a plan step for π_k is similar to the previous case.

This concludes the inductive proof for claims (1) and (2). As mentioned above, applying Theorem 3.14 to this result permits to conclude that $A(\pi_n)$ is a solution for $\mathbb{M}_{\mathsf{Ag}}$. ∎

Proof of Theorem 4.12 (Completeness of the dialogue-based planning algorithm).

Proof. From the assumption that a $\subseteq$-minimal solution A' exists, i.e. $G \subseteq \mathsf{warr}(\mathbb{P}_{\mathsf{Ag}} \diamond A')$, we first proceed as in the proof of the Completeness Theorem 3.15 (now for the planning domain $\mathbb{M}_{\mathsf{Ag}}$). Again, from the set of actions A', we obtain the sets $\mathsf{Lines}, \mathsf{Steps}, \mathsf{Threats}$, and also a sequence $\pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ where $\mathsf{Steps} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. Using Theorem 3.15, we know that $A({\pi_n}^{\mathbb{M}_{\mathsf{Ag}}}) = A'$ and $\mathsf{goals}({\pi_n}^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$ and $\mathsf{threats}({\pi_n}^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$.

From this and the initial fact that $\pi_{\varnothing} \in \mathsf{Trueplans}_0$, the next two claims can easily be shown by induction on k:

(1) for each $k < n$ and turn m_k such that $\pi_k \in \mathsf{Trueplans}_{m_k}$, there exists a finite $m' > m_k$ such that $\pi_k(\mathcal{A}_{k+1}) \in \mathsf{Plans}_{m'}$;

(2) for each $k \leq n$ and turn $m' > m_k$ such that $\pi_{k+1} \in \mathsf{Plans}_{m'}$, there exists a finite $m_{k+1} > m'$ such that $\pi_{k+1} \in \mathsf{Trueplans}_{m_{k+1}}$.

It only remains to check that π_n satisfies the Terminating Condition for $\mathbb{M}_{m_n}$. But this follows from the facts $\mathsf{goals}(\pi_n^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$ and $\mathsf{threats}(\pi_n^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$ (in Theorem 3.15)

together with the properties of $\mathsf{Trueplans}_{m_n}$ (Theorem 4.10)

$$\mathsf{goals}(\pi_n{}^{\mathbb{M}^{m_n}}) = \mathsf{goals}(\pi_n^{\mathbb{M}_{\mathsf{Ag}}}) \qquad \mathsf{threats}(\pi_n{}^{\mathbb{M}^{m_n}}) = \mathsf{threats}(\pi_n^{\mathbb{M}_{\mathsf{Ag}}})$$

Thus, the dialogue-based algorithm will terminate with an output, e.g. a plan π_n satisfying $A(\pi_n) = A'$, if no other solution has already been found by it. ∎

Acknowledgments

This research was partially supported by the projects Epistemic Protocol Synthesis (FFI2011-15945-E), EdeTRI (TIN2012-39348-C02-01) and AT (CONSOLIDER-INGENIO 2010, CSD2007-00022), funded by the MINECO Spanish ministry. We would also like to thank the anonymous reviewers for their helpful suggestions.

References

[1] L. Amgoud, A formal framework for handling conflicting desires. In *Proc. of ECSQARU 2003*, LNAI 2711, pp. 552–563 (2003)

[2] J. Augusto and G. Simari. Temporal Defeasible Reasoning. *Knowledge and Information Systems*, 3:287–318 (2001)

[3] A. Belesiotis, M. Rovatsos, and I. Rahwan. Agreeing on plans through iterated disputes. In *Proc. of AAMAS 2010*, pp. 765–772 (2010)

[4] M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms, *Artificial Intelligence*, 171:286–310 (2007)

[5] L. Cobo, D. Martínez and G. Simari. Stable Extensions in Timed Argumentation Frameworks. In *Proc. of TAFA 2011*, S. Modgil et al (eds.), LNCS 7132, pp. 181–196 (2012)

[6] P. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, *Artificial Intelligence*, 77(2): 321–357 (1995)

[7] D. García *Planificación y Formalización de Acciones para Agentes Inteligentes*, PhD thesis, Universidad Nacional del Sur (2011)

[8] D. García and A. García and G. Simari *Defeasible Reasoning and Partial Order Planning*, Proc. of *FoIKS 2008*, LNCS 4932, pp. 311–328 (2008)

[9] A. García and G. Simari. Defeasible logic programming: An argumentative approach, *Theory and Practice of Logic Programming*, 4(1+2): 95–138 (2004)

[10] M. Ghallab, D. Nau and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, San Francisco, USA (2004)

[11] J. Hulstijn and L. van der Torre. Combining goal generation and planning in an argumentation framework. In Proc. of the *10th NMR* (NMR'04), pp. 212–218 (2004)

[12] R. Medellin-Gasque, K. Atkinson, P. McBurney, and T. Bench-Capon. Arguments over cooperative plans. In Proc. of *TAFA 2011*, S. Modgil et al (eds.), LNCS 7132, pp. 50–66 (2012)

[13] S. Pajares, E. Onaindia. Defeasible argumentation for multi-agent planning in ambient intelligence applications. In Proc. of *AAMAS 2012*, pp. 509–516 (2012)

[14] S. Pajares and E. Onaindia. Context-Aware Multi-Agent Planning in Intelligent Environments, *Information Sciences*, 227:22–42 (2013)

[15] P. Pardo and L. Godo. t-DeLP: an argumentation-based Temporal Defeasible Logic Programming framework, *Annals of Math. and Artif. Intel.*, 69(1):3–35 (2013)

[16] P. Pardo *Logical planning in temporal defeasible and dynamic epistemic logics: the case of t-DeLP and LCC*, PhD thesis, Universitat de Barcelona (2013)

[17] P. Pardo and L. Godo. An argumentation-based multi-agent temporal planning system built on t-DeLP. In Proc. of *CAEPIA 2013*, LNAI vol. 8109, pp. 188–198 (2013)

[18] P. Pardo and L. Godo. A temporal argumentation approach to cooperative planning using dialogues. In Proc. *CLIMA XIV*, LNAI vol. 8143, pp. 307–324 (2013)

[19] P. Pardo, S. Pajares, E. Onaindia, L. Godo and P. Dellunde. Multiagent argumentation for cooperative planning in DeLP-POP. In Proc. *AAMAS 2011*, pp. 971–978 (2012)

[20] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124 (2010)

[21] I. Rahwan and L. Amgoud, An Argumentation-Based Approach for Practical Reasoning, In Proc. of *ArgMAS 2006*, N.Maudet et al. (eds.), LNAI 4766, pp. 74–90 (2007)

[22] G. Simari and R. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial intelligence*, 53:125–157 (1992)

[23] F. Stolzenburg, A. García, C. Chesñevar and G. Simari. Computing Generalized Specificity. *Journal of Applied Non-Classical Logics*, 12(1):1–27 (2002)

[24] Y. Tang, T. Norman, and S. Parsons. A model for integrating dialogue and the execution of joint plans. In Proc. of *ArgMAS 2009*, LNAI 6057, pp. 60–78 (2010)

[25] A. Torreño, E. Onaindia and O. Sapena. An approach to multi-agent planning with incomplete information, Proc. of *ECAI 2012*, IOS Press, pp. 762–767 (2012)