Defeasible Argumentation for Cooperative Multi-Agent Planning

Sergio Pajares Ferrando

Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

A thesis submitted for the degree of

Doctor of Philosophy in the subject of Computer Science
Under the supervision of:
Dr. Eva Onaindia de la Rivaherrera

November 2015

PhD Thesis

Title: Defeasible Argumentation for Cooperative Multi-Agent Planning

Author: Sergio Pajares Ferrando

Advisors: Dr. Eva Onaindia de la Rivaherrera

Abstract

Abstract

Multi-Agent Systems (MAS), Argumentation and Automated Planning are three lines of investigations within the field of Artificial Intelligence (AI) that have been extensively studied over the last years. A MAS is a system composed of multiple intelligent agents that interact with each other and it is used to solve problems whose solution requires the presence of various functional and autonomous entities. Multi-agent systems can be used to solve problems that are difficult or impossible to resolve for an individual agent. On the other hand, Argumentation refers to the construction and subsequent exchange (iteratively) of arguments between a group of agents, with the aim of arguing for or against a particular proposal. Regarding Automated Planning, given an initial state of the world, a goal to achieve, and a set of possible actions, the goal is to build programs that can automatically calculate a plan to reach the final state from the initial state.

The main objective of this thesis is to propose a model that combines and integrates these three research lines. More specifically, we consider a MAS as a team of agents with planning and argumentation capabilities. In that sense, given a planning problem with a set of objectives, (cooperative) agents jointly construct a plan to satisfy the objectives of the problem while they defeasibly reason about the environmental conditions so as to provide a stronger guarantee of success of the plan at execution time. Therefore, the goal

is to use the planning knowledge to build a plan while agents beliefs about the impact of unexpected environmental conditions is used to select the plan which is less likely to fail at execution time. Thus, the system is intended to return collaborative plans that are more robust and adapted to the circumstances of the execution environment.

In this thesis, we designed, built and evaluated a model of argumentation based on defeasible reasoning for planning cooperative multi-agent system. The designed system is independent of the domain, thus demonstrating the ability to solve problems in different application contexts. Specifically, the system has been tested in context sensitive domains such as Ambient Intelligence as well as with problems used in the International Planning Competitions.

Resumen

Dentro de la Inteligencia Artificial (IA), existen tres ramas que han sido ampliamente estudiadas en los últimos años: Sistemas Multi-Agente (SMA), Argumentación y Planificación Automática. Un SMA es un sistema compuesto por múltiples agentes inteligentes que interactúan entre sí y se utilizan para resolver problemas cuya solución requiere la presencia de diversas entidades funcionales y autónomas. Los sistemas multiagente pueden ser utilizados para resolver problemas que son difíciles o imposibles de resolver para un agente individual. Por otra parte, la Argumentación consiste en la construcción y posterior intercambio (iterativamente) de argumentos entre un conjunto de agentes, con el objetivo de razonar a favor o en contra de una determinada propuesta. Con respecto a la Planificación Automática, dado un estado inicial del mundo, un objetivo a alcanzar, y un conjunto de acciones posibles, el objetivo es construir programas capaces de calcular de forma automática un plan que permita alcanzar el estado final a partir del estado inicial.

El principal objetivo de esta tesis es proponer un modelo que combine e integre las tres líneas anteriores. Más específicamente, nosotros consideramos un SMA como un equipo de agentes con capacidades de planificación y argumentación. En ese sentido, dado un problema de planificación con un conjunto de objetivos, los agentes (cooperativos) construyen conjuntamente un plan para resolver los objetivos del problema y, al mismo tiempo, razonan sobre la viabilidad de los planes, utilizando como herramienta de diálogo la Argumentación. Por tanto, el objetivo no es sólo obtener automáticamente un plan solución generado de forma colaborativa entre los agentes, sino también utilizar las creencias de los agentes sobre la información del contexto para razonar acerca de la viabilidad de los planes en su futura etapa de ejecución. De esta forma, se pretende que el sistema sea capaz de devolver planes colaborativos más robustos y adaptados a las circunstancias del entorno de ejecución.

En esta tesis se diseña, construye y evalúa un modelo de argumentación basado en razonamiento *defeasible* para un sistema de planificación cooperativa multiagente. El sistema diseñado es independiente del dominio, demostrando así la capacidad de resolver problemas en diferentes contextos de aplicación. Concretamente el sistema se ha evaluado en dominios sensibles al contexto como es la Inteligencia Ambiental y en problemas de las competiciones internacionales de planificación.

Resum

Dins de la intel·ligència artificial (IA), hi han tres branques que han sigut àmpliament estudiades en els últims anys: Sistemes Multi-Agent (SMA), Argumentació i Planificació Automàtica. Un SMA es un sistema compost per múltiples agents intel·ligents que interactúen entre si i s'utilitzen per a resoldre problemas la solución dels quals requereix la presència de diverses entitats funcionals i autònomes. Els sistemes multiagente poden ser utilitzats per a resoldre problemes que són difícils o impossibles de resoldre per a un agent individual. D'altra banda, l'Argumentació consistiex en la construcció i posterior intercanvi (iterativament) d'arguments entre un conjunt d'agents, amb l'objectiu de raonar a favor o en contra d'una determinada proposta. Respecte a la Planificació Automàtica, donat un estat inicial del món, un objectiu a aconseguir, i un conjunt d'accions possibles, l'objectiu és construir programes capaços de calcular de forma automàtica un pla que permeta aconseguir l'estat final a partir de l'estat inicial.

El principal objectiu d'aquesta tesi és proposar un model que combine i integre les tres línies anteriors. Més específicament, nosaltres considerem un SMA com un equip d'agents amb capacitats de planificació i argumentació. En aquest sentit, donat un problema de planificació amb un conjunt d'objectius, els agents (cooperatius) construeixen conjuntament un pla per a resoldre els objectius del problema i, al mateix temps, raonen sobre la viabilitat dels plans, utilitzant com a ferramenta de diàleg l'Argumentació. Per tant, l'objectiu no és només obtindre automàticament un pla solució generat de forma col·laborativa entre els agents, sinó també utilitzar les creences dels agents sobre la informació del context per a raonar sobre la viabilitat dels plans en la seua futura etapa d'execució. D'aquesta manera, es pretén que el sistema siga capaç de tornar plans col·laboratius més robustos i adaptats a les circumstàncies de l'entorn d'execució.

En aquesta tesi es dissenya, construeix i avalua un model d'argumentació basat en raonament *defeasible* per a un sistema de planificació cooperativa multiagent. El sistema

dissenyat és independent del domini, demostrant així la capacitat de resoldre problemes en diferents contextos d'aplicació. Concretament el sistema s'ha avaluat en dominis sensibles al context com és la inte l·ligència Ambiental i en problemes de les competicions internacionals de planificació.

To my family.

Without you dedication and support, this thesis would not be possible.

Acknowledgements

It is incredibly hard to thank all of the people that, in some way or another, have helped me during these graduate years.

First of all, I would like to thank Eva Onaindia for her patience. It is difficult for me to express everything I've learned from Eva during these years. Really thanks.

I would like to thank Irina Dionisio. She only knows all the time that I have been working at the computer to advance this thesis. It's amazing the patience that she has had with me. Without her support it would not have been able to complete this thesis. Thanks also to Pablo, who I still do not know him but I'm sure that he will be the second love of my life. Thanks also to my cat Coco for its unconditional love.

I would like to thank my parents, my sister and brother, since they have been there for me whenever I needed them.

It would be inconsiderate if I forgot about my labmates. The moments of fun and joy have been uncountable. Thanks to Sergio E., Alejandro, Víctor, César, Pablo, Sonia, Clara, etc.

Finally, thank all the people who in one way or another have helped me to achieve this thesis.

Contents

List of Figures

Abbreviations and Acronyms

AI: Artificial Intelligence.

MAPOP: Multi-Agent Partial Order Planning.

CAMAP: Context-Aware Multi-Agent Planning.

MAPA: MultiAgent Planning and Argumentation.

MAP: Multi-Agent Planning.

DeLP: Defeasible Logic Programming.

POP: Partial Order Planning.

TMAP: Traditional Multi-Agent Planning.

DeLP-POP: A POP planner based on DeLP argumentation.

DeLP-MAPOP: A extension of DeLP-POP to multi-agent systems.

Q-DeLP-POP: Framework that extends DeLP-POP for dealing with the *qualification problem*.

Q-DeLP-MAP: Extension of Q-DeLP-POP to multi-agent systems.

PS-Q-DeLP-MAP: Adaptation of Q-DeLP-MAP for using argumentation to select between solution plans.

IPC: International Planning Competition.

DefPlanner: argumentation-based partial-order planner.

STRIPS: Stanford Research Institute Problem Solver.

PDDL: Planning Domain Definition Language.

AmI: Ambient Intelligence.

1

Introduction

The present document is not organized as a conventional PhD dissertation but as a collection of the papers derived from the development of the work. This is a PhD modality accepted by our university whenever the publications allows to capture the main contributions of the doctoral work. We believe this is the case and so we have opted for this option.

Nevertheless, this document is not simply a list of publications. This chapter, in particular, aims at guiding the reader through the contents of the document. Thus, in section 1.1 we present the motivation that originated this PhD thesis through a brief tour along the state-of-the-art in planning and argumentation. Section 1.2 shows the main objectives of the work, section 1.3.1 presents the list of scientific publications and we summarize the research projects which this PhD dissertation has contributed to in section 1.3.2. The other two sections are intended to be a guide for the reader throughout the remainder of the document.

1.1 Motivation

One common problem in Artificial Intelligence (AI) is to select the best course of action for an agent; i.e, reasoning about *what to do*. This problem has been primarily addressed from two standpoints: the knowledge or epistemological perspective, which puts the emphasis on the representation of the world such that the solution of a problem follows from the representation, and the reasoning or heuristic perspective, mostly concerned with the information for solving the problem and the reasoning process on an abstract and formal representation of the world (1). *Practical reasoning*, the research line mostly focused on the epistemological view, includes a great deal of epistemic reasoning, directed at determining what to believe (2). *Automated planning*, on the other hand, is concerned with the computational process for the selection and organization of the actions. Back in the 90's, Pollock concluded that since epistemic cognition is defeasible, a planning agent must be prepared to revise its plans as its defeasibly held beliefs change and may have to acquire more information through reasoning to solve a planning problem (3).

Figure 1.1 outlines our view of these two approaches to address the problem of selecting the best course of action for an agent, which are explained in detail below.

The mainstream in *practical reasoning* lies in the use of argumentation theory so as to extend the means-end reasoning in classical planning with presumptive justifications for the adoption of a particular action. Particularly, *practical reasoning* aims at a methodological approach for reasoning about actions that draws upon the principles of reasoning about beliefs. Using Dung's argumentation framework over beliefs (4) has been the predominant approach in *practical reasoning*. The research by Rahwan and Amgoud in (5) proposes a framework for arguing about what desires an agent should adopt and one for arguing about what plans to intend in order to achieve these desires on the basis of Dung's abstract argumentation theory. Instantiations of Dung's argumentation framework have also been used to study the goal deliberation process (6) or the generation of consistent

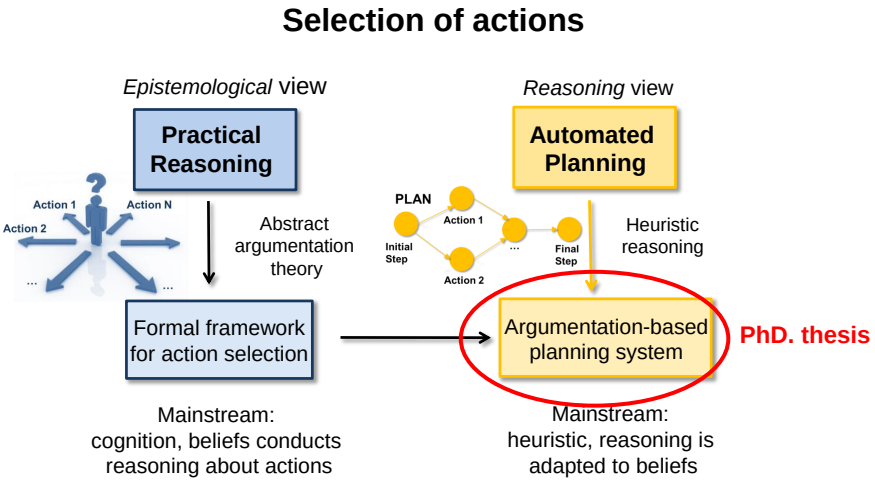

Figure 1.1: Selection of actions from the *practical reasoning* and *automated planning* standpoints.

plans from a set of conflicting beliefs (7). Building argumentation plans for negotiating conflict resolution at a planning stage is also an interesting application of argumentation in *practical reasoning* (8).

Some other works, however, follow the notion of argument scheme proposed by Walton (9) and present an approach in which arguments and conflicts are represented as argument schemes and critical questions, respectively (10, 11). This approach represents preferences based upon individual values, thus enabling argumentation be evaluated as a Value-Based Argumentation Framework (12), an extension to Dung's argumentation framework. The proposal in (10) has been one of the most popular approaches in *practical reasoning*, it has demonstrated its applicability in domains such as law, experimental economics or e-democracy (13, 14, 15) and it has also been exploited for the design of argumentation-based dialogues to support automated coordination in distributed planning

(16), multi-agent deliberation dialogues (17) or the construction of joint plans (18).

In general, the research in the context of *practical reasoning* is aimed at proposing a formal framework for action selection under the statement that *'what I believe conducts reasoning about actions'* but it usually neglects the problem-solving process required when dealing with planning problems. Unlike argumentation-based approaches of *practical reasoning*, another line of investigation closer to *automated planning* also explores the relationships between classical planning and argumentation but building upon a planning formalism and using argumentation to guide the reasoning process. A first step in this direction pointed by Pollock assumes that agent's deductions are not always certain information, but *plausible*, and the conclusions can be withdrawn when new pieces of knowledge are found; i.e., agents must use defeasible reasoning (19). OSCAR is a goal-regression planner that essentially performs the same search of Partial-Order Planning (POP) but reasoning defeasibly about candidate plans at the end of the planning process (20). In OSCAR, the plan search itself is also done defeasibly, thus enabling to reason about the impact of unexpected environmental conditions on the solution plans and to select the plan which is less likely to fail at execution time. All in all, the OSCAR architecture for rational agents is one of the first attempts to build a planner based upon a defeasible reasoner.

In the same line, another pioneer work on using argumentation in planning was proposed by Ferguson and Allen in (21). The authors of this work present a formal model of plans based on a defeasible argument system that is able to suggest aspects of a plan, criticize a plan and revise the plan (21). Both investigations in (20) and (21), considered as the first steps towards building an *argumentation-based planning* system, have clear similarities to the works on plan modification and replanning but rather than forcing the planner to resort to replanning in light of new information, they consider planning within the context of a general defeasible reasoning system. That is, they adapt the reasoning

about actions to the beliefs about the environment (this is precisely the investigation line of this PhD thesis, which is shown in red in Figure 1.1).

More recently, Simari et al present a defeasible argumentation framework for the definition of actions and the combination of these actions into plans (22). This work lays the foundations of an argumentation-based formalism for constructing plans (23) by using Defeasible Logic Programming (DeLP) (24), a formalism to represent the knowledge and build applications that deal with incomplete and contradictory information in dynamic domains. The formalism presented in (23, 25), which we will refer to as DeLP-POP in the following, analyzes the interplay of arguments and actions when constructing plans using POP techniques.

Subsequently, further investigations on argumentation-based planning focus on the application of argument-based systems to Multi-Agent Planning (MAP). Our proposal presented in (26) is a formal extension of DeLP-POP to a multi-agent context, in which agents are assumed to have planning and argumentation capabilities. Specifically, this work proposes a formal dialogue for argumentative plan search, by which agents exchange plan proposals and arguments for or against such proposals. A first practical extension for evaluating the approach in (26) in ambient intelligent applications is presented in (27, 28). These two latter works also contribute with a modelization language for defeasible rules within a planning context.

An argumentation-based dialogue protocol that enable agents to discuss candidate plans and reach agreements was proposed in (29, 30), where candidate plans of the agents are generated by an external single-agent planner and the protocol is used then for reasoning about the contradictory planning beliefs in the candidate plans and select a valid solution plan. In this approach, agents use argumentation to defend or attack the candidate plans put forward by others, not for cooperatively building a plan contributed by several agents. Another interesting work that combines the benefits of argumentation in MAP emphasizes the utilization of argumentation to solve conflicts between sub-plans

of different agents by means of deliberative dialogues based on argumentation schemes (31, 32). Conflicts may be caused by concurrent actions, plan constraints or norms the agents must adhere to and argumentation is used as a mechanism for analyzing these conflicts when several sub-plans of different agents have to be merged. Likewise, argumentation in (31, 32) is not used for building plans but for arguing at end of the planning generation, where the planning task is accomplished by an external classical planner. A different approach that also makes use of argumentation schemes proposes structured argumentative dialogues to coordinate plan-related tasks (16). Particularly, in this approach agents coordinate their beliefs and intentions with the use of a strategy based on an argumentation scheme and a set of related critical questions for selecting plan proposals. Thus, choosing an appropriate question in the dialogue becomes an important issue in terms of dialogue and cooperation efficiency.

Most of the existing argumentation-based MAP models aim at using argumentation for solving conflicts within a plan or solving conflicts among plans proposed by different agents, like the works in (16, 29, 30, 31, 32). In these formalisms, a plan is generated by an external classical planner, which is then transformed accordingly so as to be used in argumentative dialogues. Unlike some of the initial single-agent planning approaches that use defeasible reasoning during the plan search (22, 24), these MAP models opt for arguing about tentative solution plans rather than arguing the reasoning process that lead to these solution plans. One glaring reason for adopting this trend is that applying defeasible reasoning during the process of solving a MAP problem requires developing a multi-agent planner, which is a complex and arduous task.

The line of investigation of this PhD thesis lies in the defeasible construction of plans for solving MAP problems. We leverage the experience of our research group on the field of multi-agent planning and, particularly, on MAP-POP (33), one of the multi-agent planners developed within the GRPS-AI group[1] and which was used as the basis for the design

[1] http://users.dsic.upv.es/grupos/grps/

and implementation of our argumentation-based MAP proposal. The foundations of the architecture of our model (27, 34) were precisely inspired by the DeLP-POP formalism (24) and the cooperative distributed planning framework presented in (33).

All in all, this PhD thesis contributes with the design, construction and evaluation of an argumentation-based MAP model for solving problems where a team of agents, which have diverse abilities and beliefs, are committed to achieve a set of goals. Agents put forward their (partial) plan proposals and engage in a stepwise dialogue consisting of exchanging arguments for or against the plan proposals. In our approach, arguments are the mechanism to take into account the context conditions during the plan search process. We thus come up with an operative, domain-independent and fully-integrated argumentation and planning framework for multi-agent contexts. To the best of our knowledge, our proposal is the first defeasible planner capable to tackle MAP problems from the International Planning Competitions (IPC)[1] (35).

1.2 Objectives

In this section, we present the objectives that motivated this PhD dissertation through a tour which shows the different lines of investigation covered in the work and the pursued stages to reach each milestone. Then, in section 1.2.1, we present the problem formulation and we provide a more precise overview of the paradigms and techniques that we chose to carry out our objectives.

This work revolves around the problem of **Multi-Agent Planning** and, more specifically, on cooperative planning, wherein multiple entities (agents) work cooperatively in order to build a plan that solves a set of common goals. Cooperative MAP is currently a hot research line in which our group is actively working. Under this paradigm, agents are interpreted as entities that have different planning capabilities (specialized agents) or

[1] http://ipc.icaps-conference.org/

entities that are geographically distributed and do not have access to all the data of the world. These different world views and planning capabilities of the agents define their vision of the planning task.

In classical approaches to MAP, agents only work with the information defined in the planning task: facts that describe the initial state of the world and a set of actions that model the agents abilities. Then, agents realize a search process by which they infer new facts that respond to the dynamics and causal relationships defined in the planning task. That is, classical planning approaches assume that the planner (agent) begins with all the relevant, accurate and necessary knowledge for solving the problem. However, in complex and dynamic environments, encoding the complete know-how knowledge of a planning agent into the planning task is not affordable and traditional planning systems are not capable of properly exploiting this know-how knowledge that goes beyond the factual knowledge and causal reasoning. Thus, our principal objective is to extend a traditional MAP task so that the non-factual individual knowledge of the agents is considered when building a plan. This new source of information, which stems from the expertise and beliefs of the agents but cannot be regarded as a universal truth, will be modeled as **defeasible knowledge** which is susceptible of change in light of new information provided by other agents.

The goal of planning is choosing the proper actions to achieve the task goals according to some optimization criteria. This is done by using the factual information and known abilities of the agents. The purpose of introducing defeasible knowledge in a planning task is to account for all the external conditions that might potentially affect the execution of an action in the real world beyond the conditions defined in the causal model. In order to handle the agents' defeasible knowledge that will be used to conduct the planning process towards a successful plan execution, we opted by a **multi-agent argumentation model**. Planning agents will use their know-how knowledge and beliefs of the world to argue about the context in which an action will be executed and to prevent the action from

a potential execution failure. In certain domains, it is crucial to obtain robust plans; i.e., plans that are executable despite uncertainty in the execution environment. For example, given an emergency situation where an ambulance has to be sent to the patient's home, a traditional MAP approach would not take into account the contextual conditions that are not expressed in the causal theory (e.g., traffic jam), and so the solution plan is likely to fail during its execution in the real world.

Our argumentation model draws upon a defeasible knowledge reasoning by which agents argue about the tentative contextual conditions that might prevent an action from executing successfully. From the planning standpoint, the goal is to build plans consistent with the causal theory that defines the dynamics of the world. From the argumentation standpoint, the goal is to obtain plans consistent with the agents beliefs so as to increase the likelihood of achieving a robust plan. Thereby, the quality of the plans will be assessed according to the usual planning quality criteria (number of actions and plan duration) as well as by their level of robustness (minimization of the risk of unexpected action failures).

Ultimately, the aim of this thesis is to design, build and evaluate a **domain-independent argumentation-based** MAP model to solve planning problems where the know-how knowledge of the agents in the context of application is integrated in the planning process.

1.2.1 Problem formulation and solution specification

In this section, we present the problem we want to solve, the proposed steps to tackle the problem resolution and the solution specification.

As commented before, the starting point of this work is the definition of a cooperative MAP task. We believe that partial-order planning (POP) is the paradigm that better fits the multi-agent nature of cooperative planning wherein various agents need to work together for solving the problem. Therefore, we adopt the MAP task definition proposed in the

MAP-POP approach (33), defined as a tuple $\langle \mathsf{AG}, V, A, \Psi, G \rangle$, where AG is a finite and non-empty set of planning agents, V is the set of state variables managed by AG, A is the set of actions of the agents, Ψ is the initial state of the problem and G is the set of goals to be achieved by AG. Two or more agents can share actions of A and can share pairs variable-value from Ψ.

Given that our objective is to test the resulting model in problems of the IPC benchmarks, we will use the so well-known PDDL language problem specification adopted by the planning community. Particularly, we will assume the PDDL codification of the multi-agent planning tasks presented by MAP-POP.

The next step is to choose a formalism to represent the defeasible knowledge and integrate it within the MAP task definition. The Defeasible Logic Programming (DeLP) formalism combines Logic Programming and Defeasible Argumentation (24) and provides the possibility of representing information in the form of rules in a declarative manner, and a defeasible argumentation inference mechanism for warranting the entailed conclusions. The declarative nature of the knowledge represented in DeLP fits very well with the declarative planning knowledge represented in PDDL. In DeLP, the effect of a defeasible rule comes from a dialectical analysis, made by the inference mechanism, which involves the consideration of arguments and counter-arguments where that rule is included. Thus, our purpose is to integrate the defeasible reasons of an agent to believe something within the POP paradigm and use these beliefs to argue or support the inclusion of a action in the plan. This implies to establish a clear distinction between the planning knowledge that defines the causal theory of the domain and the know-how and defeasible knowledge of the agents. More specifically, we will analyze the framework DeLP-POP, an extension of POP with DeLP-style argumentation for single-agent planning, where both actions and arguments may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. Our principle of design is to introduce arguments not only to intentionally support some step of a plan but also to defeat

or defend other arguments in the plan. This principle of design is in line with DeLP-POP but our final purpose is to extend this approximation to a multi-agent context.

Integrating defeasible knowledge within a multi-agent partial-order planner implies some significant modifications in the POP machinery. First, the definition of a MAP task must now include the defeasible knowledge of the agents $\langle \mathsf{AG}, V, A, \Psi, G, \Delta \rangle$, where Δ represents the set of defeasible rules of AG. Second, arguments will be deduced from Δ and Ψ. Third, since arguments will be used by the agents to support or contradict the inclusion of an action in the plan, we need a representational scheme that allows us to handle actions as an element of a partial-order plan as well as a piece of defeasible knowledge. That is, the model must be able to represent, handle and reason with A and Δ jointly. More specifically, we need to represent the actions contributed by the agents to the plan in such a way that any other agent can argue or support the applicability of the action under the environment conditions. This will eventually lead to a novel proposal of addressing the qualification problem.

The operational framework of our proposal requires to design and build multi-agent dialogues so that each step of a partial plan can be discussed among agents. Specifically, we will design a dialogue mechanism by which agents exchange arguments about the conditions that might affect the feasibility of an action in the real world according to their know-how knowledge and beliefs. Therefore, agents will not only be equipped with planning capabilities, but also with argumentation abilities.

Our goal is to obtain more robust plans than the plans returned by MAP-POP; i.e., solutions that demonstrate that incorporating the know-how knowledge of the agents in the form of defeasible knowledge allows us to obtain more robust executable plans. For this purpose agents will instantiate argumentative dialogues, while planning, to reason about each step comprised in the plan.

Finally, we aim for a domain-independent, fully integrated and operative argumentation-based MAP model. This means we have not only to design and implement the model but

also test it in different applications domains. Specifically, we will validate the system in applications of ambient intelligence in the field of health-care as well as in problems from the IPC benchmarks. We will carry out several experiments considering various levels of difficulty of the planning and argumentation problems.

1.3 Scientific Contributions

In this section, we present the list of contributions related to this PhD thesis..

1.3.1 Scientific Publications

In this section, we show the list of publications related to this PhD thesis.

1.3.1.1 Selected Papers

1. **Selected Paper 1:** Defeasible Planning through Multi-Agent Argumentation (CIMA-ICTAI 2010 and its extension in a Book):

    - S. Pajares and E. Onaindía. **DefPlanner: A defeasible argumentation-based planner.** 2nd International Workshop on Combinations of Intelligent Methods and Applications (CIMA'10), in conjunction with 22th International Conference on Tools with Artificial Intelligence (ICTAI 2010) pp. 34-42. (2010).
    - S. Pajares and E. Onaindía. **Defeasible Planning through Multi-Agent Argumentation.** Combinations of Intelligent Methods and Applications, Smart Innovation, Systems and Technologies Vol. 8 pp. 1-19. (2011).

2. **Selected Paper 2:** An architecture for Defeasible-Reasoning-based Cooperative Distributed Planning (CoopIS 2011):

- S. Pajares, E. Onaindía and A. Torreño. **An architecture for Defeasible-Reasoning-based Cooperative Distributed Planning.** 19th International Conference on Cooperative Information Systems (CoopIS 2011) pp. 200-217. (2011).

3. **Selected paper 3:** Multiagent Argumentation for Cooperative Planning in DeLP-POP (AAMAS 2011) and its extension in (ArgMAS 2012):

   - P. Pardo, S. Pajares, E. Onaindía, L. Godo and P. Dellunde. **Multiagent Argumentation for Cooperative Planning in DeLP-POP.** 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011) pp. 971-978. (2011).
   - P. Pardo, S. Pajares, E. Onaindía, L. Godo and P. Dellunde. **Cooperative Dialogues for Defeasible Argumentation-based Planning.** Argumentation in Multi-Agent Systems (ArgMAS). Vol. 7543, pp. 185-204. (2012).

4. **Selected paper 4:** Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications (AAMAS 2012):

   - S. Pajares and E. Onaindía. **Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications.** 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012) pp. 509-516. (2012).

5. **Selected Paper 5:** Context-Aware Multi-Agent Planning in Intelligent Environments (INS Journal 2013):

   - S. Pajares and E. Onaindía. **Context-Aware Multi-Agent Planning in Intelligent Environments.** Information Sciences Journal. Vol. 227 pp. 22-42.

(2013). Impact Factor 2013: 3.89 (Q1 Computer Science, Artificial Intelligence).

6. **Selected Paper 6:** Argumentation-based Planning (Submitted AIJ 2015):

   - S. Pajares and E. Onaindía. **Argumentation-based Planning.** Artificial Intelligence Journal. (Under Revision). (2015). Impact Factor 2015: 3.37 (Q1 Computer Science, Artificial Intelligence).

1.3.1.2 Other Publications

- S. Pajares and E. Onaindía. **Temporal Defeasible Argumentation in Multi-Agent Planning.** 22nd International Joint Conferences on Artificial Intelligence (IJCAI 2011) pp. 2834-2835. (2011).

1.3.2 Research Projects

The work of this thesis could have not been possible without a 4-year (only 2 of them consumed) FPU research grant granted by the Spanish Government.

In addition, this thesis has provided significant advances to the following research projects funded by the Spanish Government:

- "Agreement Technologies" Consolider-INGENIO 2010 under grant CSD2007-00022 (Main Researcher: Carles Sierra, from 2007 to 2012). Agreement technologies is a term coined in the last few years to refer to those technologies that allow computational entities to automatically solve conflicts. Being used by humans so frequently, argumentation-based planning is one of the key technologies in agreement technologies. The work carried out in this thesis aims to advance the state-of-the-art in argumentation mechanisms during the planning process.

- "Magentix2: A Multi-agent Platform for Open Multi-agent Systems" under grant TIN2008-04446 (Main Researcher: Ana Garcia-Fornes, from 2008 to 2011). Magentix2 is a multi-agent platform. The work of this thesis aims to provide planning and argumentation capabilities for Magentix2 agents.

- "Multi-agent Plan Interaction" under grant TIN2011-27652-C03-01-AR (Main Researcher: Eva Onaindia, from 2012). The work carried out in this thesis aims to provide an argumentation-based multi-agent planner.

Additionally, this thesis has also provided advances to "Advances on Agreement Technologies for Computational Entities" PROMETEO/2008/051 (Main Reseacher: Vicente Botti) funded by the Valencian Government.

1.4 Document Structure

The remainder of this thesis is structured as follows.

- **Chapter 2. Selected Papers:** This chapter consists on a collection of the main articles (conferences and journals) published by the PhD. student which support this thesis. These articles have been completely integrated into the present document of thesis, so there is only one section of Bibliographical References at the end of this document (thus avoiding repeated references between articles).

- **Chapter 3. General Discussion of the Results:** This chapter provides a discussion about all the obtained results presented in the articles from the previous chapter.

- **Chapter 4. Conclusions:** This last chapter is devoted to present a final review of the conclusions as well as promising directions for further works.

2

Selected Papers

This chapter compiles the most relevant research papers published during the development of this PhD thesis. The articles are chronologically listed and provide a thorough description of the scientific contributions that conform this PhD thesis. This chapter is organized as follows: section 2.1 describes the articles included in this chapter and briefly summarizes their contents. The subsequent sections include the full text of the research articles adapted to the format of the present PhD thesis.

2.1 Summary of the Selected Papers

The results obtained during the development of the present PhD thesis have been systematically communicated through the publication of a wide range of scientific papers. This chapter focuses on the impact articles that synthesize the main body of work of this PhD thesis, offering a clear and comprehensive summary of the results obtained.

The next sections arrange these scientific papers according to their date of publication, which gives a clear idea of the evolution of the research activities and the main milestones reached during the development of this PhD thesis.

Firstly, section 2.2 presents the first two works of this thesis. Firstly, it was published in the 2nd International Workshop on Combinations of Intelligent Methods and Applications (CIMA 2010), in conjunction with 22th International Conference on Tools with Artificial Intelligence (ICTAI 2010). Secondly, the work was extended in a chapter of Combinations of Intelligent Methods and Applications, Smart Innovation, Systems and Technologies book, Vol 8. This work is an entirely theoretical approach. It introduces DefPlanner, an argumentation-based partial order planner where different agents that have a partial, and possibly contradictory, knowledge of the world articulate arguments for and against supporting preconditions of the actions to be included in a plan. This work is a first extension of (23) to a multi-agent system in argumentation. However, this work also consider the task of planning in a single agent environment, since only an agent of the multi-agent system can have the role of planner. The work introduces an example where agents have to reason about contextual conditions that might prevent a partial plan will fail at execution time.

Secondly, the article of section 2.3, originally published in the19th International Conference on Cooperative Information Systems (CoopIS 2011), presents a **M**ulti**A**gent **P**lanning and **A**rgumentation (MAPA) architecture based on a multiagent partial order planning paradigm using argumentation for communicating agents. Agents use an argumentation-based defeasible reasoning to support their own beliefs and refute the beliefs of the others according to their knowledge. Unlike the work in section 2.2, in MAPA, actions and arguments may be proposed by different agents to enforce some goal, if their conditions are known to apply and arguments are not defeated by other arguments applying. In order to plan for these goals, agents start a stepwise dialogue consisting of exchanges of plan proposals to satisfy this open goal, and they evaluate each plan proposal according to the arguments put forward for or against it. After this, an agreement must be reached in order to select the next plan to be refined. It is evaluated in a transit agencies scenario. Specifically, in a Transit Journey Planning Service (TJPS) (a specialized electronic search

engine), where MAPA is used to find the best route between two locations by using some means of transportation.

Thirdly, section 2.4 summarizes two works: one published in the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), and its extension in the Argumentation in Multi-Agent Systems (ArgMAS 2012). This contribution proposes a model for argumentation-based multi-agent planning, with a focus on cooperative scenarios. It consists in a multi-agent extension of DeLP-POP (23). In DeLP-POP, actions and arguments (combinations of rules and facts) may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. In a cooperative planning problem a team of agents share a set of goals but have diverse abilities and beliefs. In order to plan for these goals, agents start a stepwise dialogue consisting of exchanges of plan proposals, plus arguments against them. Since these dialogues instantiate an A^* search algorithm, these agents will find a solution if some solution exists, and moreover, it will be provably optimal (according to their knowledge). The main contribution of this work is the formalization of DeLP-POP for multi-agent systems.

Fourthly, the work of section 2.5 is published in the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), and it presents a practical extension of the theoretical model presented in section 2.4. The new framework, named DeLP-MAPOP, is implemented on a platform for open multi-agent systems and has been experimentally tested, among others, in applications of ambient intelligence in the field of health-care. DeLP-MAPOP is based on a multi-agent partial order planning paradigm in which agents have diverse abilities, use an argumentation-based defeasible reasoning to support their own beliefs and refute the beliefs of the others according to their knowledge during the plan search process. The requirements of Ambient Intelligence (AmI) environments featured by the imperfect nature of the context information and heterogeneity of the involved agents make defeasible argumentation be an ideal approach to resolve potential

conflicts caused by the contradictory information coming from the ambient agents. Moreover, the ability of AmI systems to build a course of action to achieve the user's needs is also a claiming capability in such systems. DeLP-MAPOP shows to be an adequate approach to tackle AmI problems as it gathers together in a single framework the ability of planning while it allows agents to put forward arguments that support or argue upon the accuracy, unambiguity and reliability of the context-aware information. The main contributions of this work is that is the first system in the PhD thesis that has been implemented and experimentally proven in a AmI environment.

Fifth, the work of section 2.6 is published in Information Sciences Journal, Vol. 227. This work is an extended version of the work presented in section 2.5. This article presents the specification, implementation and an exhaustive experimentation of CAMAP, a cooperative and distributed planning framework that uses defeasible argumentation to reason about the context information on smart environments. Our most relevant contribution is to come up with a fully implemented MAP framework that has been extensively tested in AmI environments, particularly on health-care applications. This this work also introduces a novel action representation which allow the model to implicitly address the qualification problem as every precondition of a planning action is now supported by an argument rather than directly by an action effect. Moreover, it presents an extension of PDDL3.1 (Planning Domain Definition Language) in order to introduce some functionalities required in a argumentation-based multi-agent planning task. Finally, it presents a more exhaustive experimentation than the presented in the work of section 2.5, in order to evaluate CAMAP in AmI environments.

Finally, the work of section 2.7 is submitted to Decision Support Systems. Now this work is under review. This paper presents Q-DeLP-POP, an argumentation-based MAP system that elaborates on two previous contributions: an initial formalization of a multi-agent argumentative planning model in the framework of DeLP-POP (26) (section 2.5), and a simple implementation of such argumentative MAP model in a domain of ambient

intelligent applications (27, 28) (sections 2.5 and 2.6). The results obtained in these latter works revealed that the argumentation-based MAP model was able to deal with rich argumentative representations but had a limited planning capability. Q-DeLP-POP, however, greatly outperforms the previous system by exploiting, among other things, the reuse of argumentative dialogues during the construction of a POP search tree, which allows us to tackle problems from the International Planning Competitions (IPC)[1]. Additionally, Q-DeLP-POP provides a more sophisticated specification of the *qualification problem* in planning, defining novel relationships between argument steps and action steps of a plan. Overall, the aim of this work was to put together and exploit the investigations carried out in (26) and (28) in order to come up with a domain-independent, fully integrated and operative argumentation-based MAP model. The experimental results of this work are considered as the final results of this PhD thesis.

Figure 2.1 shows a summary of the evolution of the contributions of this PhD thesis. The first version of this Framework was labeled as DefPlanner (section 2.2); the second version as MAPA (section 2.3); the third version as multi-agent DeLP-POP (section 2.4); the fourth version as DeLP-MAPOP (section 2.5); the fifth version as CAMAP (section 2.6); and, the sixth and final version as Q-DeLP-POP (section 2.7).

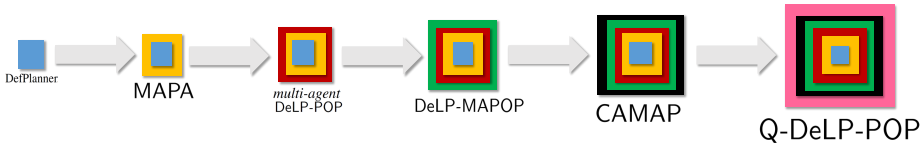

Table 2.1: The different versions of the Framework proposed in this PhD thesis.

[1] http://ipc.icaps-conference.org/

2.2 Selected Paper 1: Defeasible Planning through Multi-Agent Argumentation (CIMA-ICTAI 2010 and its extension in a Book)

Abstract. *The work reported here introduces DefPlanner, an argumentation-based partial-order planner where different agents that have a partial, and possibly contradictory, knowledge of the world articulate arguments for and against supporting preconditions of the actions to be included in a plan. In this paper, we introduce an extension to multiple agents of the defeasible argumentation formalism that has been proposed to address the task of planning in a single agent environment.*

2.2.1 Introduction

Planning is the art of building control algorithms that synthesize a course of action to achieve a desired set of goals. The mainstream in planning is that of using heuristic functions to evaluate goals and choices of action or states on the basis of their expected utility to the planning agent (36). In classical planning, intelligent agents must be able to set goals and achieve them, they have a perfect and complete knowledge of the world, and they assume their view of the world can only be changed through the execution of the planning actions. However, in many real-world applications, agents often have contradictory information about the environment and their deductions are not always certain information, but *plausible*, since the conclusions can be withdrawn when new pieces of knowledge are posted by other agents.

On the other hand, argumentation, which has recently become a very active research field in computer science (37), can be viewed as a powerful tool for reasoning about inconsistent information through a rational interaction of arguments for and against some conclusion. Systems that build on defeasible argumentation apply theoretical reasoning for the generation and evaluation of arguments, and they are used to build applications

that deal with incomplete and contradictory information in dynamic domains ((5, 24, 38, 39)). Particularly, the application of an argumentation-based formalism to deal with the defeasible nature of reasoning during the construction of a plan has been addressed by Garcia and Simari (22)(23).

This paper extends the work of (23) and presents DefPlanner, a defeasible argumentation planner developed for multi-agent environments. We explicitly consider several entities (agents) in the argumentative process for the support of the conditions of a planning action. Some recent works like (40, 41) realize argumentation in multi-agent systems using defeasible reasoning but they are not particularly concerned with the task of planning. Specifically, we consider propositional STRIPS planning representation augmented with the incorporation of different sources of defeasible information (agents). Defplanner is a partial-order planner ((42, 43)) that invokes an argumentation process where many different agents with different opinions exchange arguments and counterarguments in order to determine whether a given precondition of an action is supported or not, i.e. it can be defeasibly derived or not.

This paper is organized as follows. Next section summarizes the main notions on defeasible logic and partial-order planning. Section 2.2.3 elaborates on the use of argumentation during the construction of a partial-order plan. Section 2.2.4 presents the defeasible argumentation process in a multi-agent system, and section 2.2.5 presents an example of application. Finally, section 2.2.6 concludes and presents some future work.

2.2.2 Background

2.2.2.1 Defeasible Logic

In this section, we summarize the main concepts of the work on Defeasible Logic Programming (DeLP), a formalism that combines Logic Programming and Defeasible Argumentation (24). The basic elements in DeLP are facts and rules. Let $\mathcal{L}$ denote a set

of literals, where a literal h is a fact A or a negated fact $\sim A$, and, the symbol $\backsim$ represents the strong negation. The set of rules is divided into *strict rules*, i.e. rules encoding strict consequences, and *defeasible rules*, which derive uncertain or defeasible conclusions. A strict rule is an ordered pair $head \leftarrow body$, and a defeasible rule is an ordered pair $head \prec body$, where $head$ is a literal, and $body$ is a finite non-empty set of literals. For example, the strict rule $animal \leftarrow bird$ is denoting the piece of information "a bird is an animal". However, a defeasible rule is used to describe tentative knowledge that may be used if nothing else can be posed against it, e.g. "birds fly" (fly $\prec$ bird).

Using facts, strict and defeasible rules, an agent is able to satisfy some literal h as in other rule-based systems. Let X be a set of facts in $\mathcal{L}$, *STR* a set of strict rules, and *DEF* a set of defeasible rules. A **defeasible derivation** for a literal h from X, denoted as $X|\sim h$, consists of a finite sequence $h_1, \ldots, h_n = h$ of literals such that h_i is a fact ($h_i \in \mathcal{L}$), or there is a rule in *STR* $\cup$ *DEF* with head h_i and body $b_1, \ldots, b_k$, and every literal of the body is an element h_j of the sequence appearing before h_i ($j < i$). A set X is contradictory, denoted $X|\sim \bot$, if two contradictory literals, eg. h and $\backsim h$, can be derived from X.

In our planning framework, the agent's knowledge base is formed by a consistent set of facts Ψ, and a set of defeasible rules Δ.

Definition 1. *Let h be a literal, and let $\mathcal{K} = (\Psi, \Delta)$ be the knowledge base of an agent. We say that $\langle \mathcal{A}, h \rangle$ is an **argument structure** for h, or simply **argument** for h, if $\mathcal{A}$ is a set of defeasible rules of Δ, such that:*

- *there exists a defeasible derivation of h from $\Psi \cup \mathcal{A}$,*
- *the set $\Psi \cup \mathcal{A}$ is non-contradictory, and*
- *$\mathcal{A}$ is minimal, i.e., there is not a $\mathcal{A}' \subset \mathcal{A}$, such that $\mathcal{A}'$ satisfies the above two conditions.*

The literal h is called the conclusion of the argument, and $\mathcal{A}$ the support of the argument.

Definition 2. *Two literals h_1 and h_2 **disagree** iff the set $\Psi \cup \{h_1, h_2\}$ is contradictory. Two complementary literals h and $\sim h$ disagree because for any set Ψ, $\Psi \cup \{h, \sim h\}$ is contradictory. We say that the argument $\langle \mathcal{A}_1, h_1 \rangle$ is in conflict or counter-argues the argument $\langle \mathcal{A}_2, h_2 \rangle$ at the literal h, if and only if there exists a sub-argument $\langle \mathcal{A}, h \rangle$ of $\langle \mathcal{A}_2, h_2 \rangle$, that is $\mathcal{A} \subseteq \mathcal{A}_2$, such that h and h_1 disagree. If $\langle \mathcal{A}_1, h_1 \rangle$ is a counterargument for $\langle \mathcal{A}_2, h_2 \rangle$ at literal h, then h is called a counter-argument point, and the subargument $\langle \mathcal{A}, h \rangle$ is called the disagreement subargument (24).*

In short, two arguments are in conflict if they support contradictory conclusions, or one of the arguments is in conflict with an inner part of the other argument. That is, if the head of a defeasible rule in one of the arguments contradicts the head of a defeasible rule in the other argument.

In order to deal with counterarguments, a central aspect is to establish a formal **comparison criterion** among arguments. A possible preference relation among arguments is the so-called *generalized specificity* (44). We consider an argument $\mathcal{A}1$ is preferred to an argument $\mathcal{A}2$ if $\mathcal{A}1$ is more precise (it is based on more information), or more concise (it uses fewer rules in the conclusion derivation). In such a case, it is said $\mathcal{A}1$ is more specific than $\mathcal{A}2$. For example, $\langle \{c \multimap a, b\}, c \rangle$ is more specific than $\langle \{\sim c \multimap \sim a\}, \sim c \rangle$. We use $\langle \mathcal{A}1, h1 \rangle \succ \langle \mathcal{A}2, h2 \rangle$ to denote $\langle \mathcal{A}1, h1 \rangle$ is more specific than $\langle \mathcal{A}2, h2 \rangle$ The preference criterion is needed to decide whether an argument defeats another or not, as disagreement does not imply preference.

Definition 3. *The argument $\langle A_1, h_1 \rangle$ is a **defeater** for $\langle A_2, h_2 \rangle$ iff there is a subargument $\langle A, h \rangle$ of $\langle A_2, h_2 \rangle$ such that $\langle A_1, h_1 \rangle$ is a counterargument of $\langle A_2, h_2 \rangle$ at literal h, and $\langle A_1, h_1 \rangle \succ \langle A, h \rangle$.*

Definition 4. *An **argumentation line** for $\langle \mathcal{A}_0, h_0 \rangle$ is a sequence of arguments, denoted $\Lambda = [\langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_m, h_m \rangle]$, where each element of the sequence $\langle \mathcal{A}_i, h_i \rangle$, $i > 0$, is a defeater of its predecessor $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$. Certain constraints over Λ are considered in (24) in order to avoid several problematic and undesirable situations that may arise in Λ.*

Definition 5. *A **dialectical tree** for the argument $\langle \mathcal{A}_0, h_0 \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$, is defined by the root of the tree, labeled with $\langle \mathcal{A}_0, h_0 \rangle$, and a set of argumentation lines from the root, where every node (except the root) represents a defeater of its parent, and leaves correspond to non-defeated arguments, arguments with no defeaters.*

Some examples of dialectical trees can be found in (24). In order to decide whether the argument at the root of a given dialectical tree is defeated or not, it is necessary to perform a bottom-up analysis of the tree. Every leaf of the tree is marked *undefeated* and every inner node is marked *defeated*, if it has at least one child node marked undefeated. Otherwise, it is marked undefeated. Let $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ denote a marked dialectical tree of the argument $\langle \mathcal{A}, h \rangle$. A literal h is said to be ***warranted***, if and only if there is an argument $\langle \mathcal{A}, h \rangle$ for h such that the root of the marked dialectical tree $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ is marked undefeated. In such a case, $\langle \mathcal{A}, h \rangle$ is a warrant for h. If a literal h is a fact then h is also warranted as there are no counterarguments for $\langle \emptyset, h \rangle$. Otherwise, if all arguments for h are marked as defeated then the literal h is said to be ***not warranted***.

2.2.2.2 Partial-Order Planning

Planning is the art of building control algorithms that synthesize a course of action to achieve a desired set of goals. We consider planning problems encoded in a formal, first-order language such as STRIPS (45), particularly in a propositional version of STRIPS. We will denote the set of all propositions by $\mathcal{P}$ (ground facts or literals). A planning state s is defined as a finite set propositions $s \subseteq \mathcal{P}$. A (grounded) planning task is a triple $\mathcal{T} = \langle \mathcal{O}, \mathcal{I}, \mathsf{G} \rangle$, where $\mathcal{O}$ is the set of deterministic actions of the agent's model that describes the state changes, and $\mathcal{I} \subseteq \mathcal{P}$ (the initial state) and $\mathsf{G} \subseteq \mathcal{P}$ (the goals) are sets of propositions. An action $a \in \mathcal{O}$ is a tuple $a = (pre(a), add(a), del(a))$, where $pre(a) \subseteq \mathcal{P}$ is the set of propositions that represents the action's preconditions, and $add(a) \subseteq \mathcal{P}$ and $del(a) \subseteq \mathcal{P}$ are the sets of propositions that represent the positive and negative effects,

respectively. We will represent an action a as follows:

$$\{q_1, \ldots, q_n, \sim r_1, \ldots, \sim r_m\} \xleftarrow{id} \{p_1, \ldots, p_k\} \tag{2.1}$$

where id is the action name, $\forall_{i=1}^{k} p_i \in pre(a)$, $\forall_{i=1}^{n} q_i \in add(a)$, and $\forall_{i=1}^{m} r_i \in del(a)$. An action a is executable in state s if $pre(a) \subseteq s$. The state resulting from executing a is defined as $s' = (s \setminus del(a)) \cup add(a)$. That is, we delete any proposition in s that belongs to $del(a)$, and add the propositions in $add(a)$. A solution plan (Π) for a planning task $\mathcal{T}$ is a set of actions $\Pi = \{a_1, \ldots, a_n\} \subseteq \mathcal{O}$ such that when applied to $\mathcal{I}$, it leads to a final state in which the goals G are satisfied. A planning task $\mathcal{T}$ is solvable if there exists at least one plan for it.

In what follows, we provide a brief introduction to the Partial-Order Planning (POP) paradigm ((42)(43)). A more detailed tutorial can be found in (46). In POP, search is done through the space of incomplete partially-ordered plans as opposite to state-based planning. Thus, a key concept in POP is that of *partial-order plan*.

Definition 6. *A **partial-order plan** is a tuple $\Pi = \langle \mathcal{A}P, \mathcal{O}R, \mathcal{C}L, \mathcal{O}C, \mathcal{U}L \rangle$, where:*

- *$\mathcal{A}P \subseteq \mathcal{O}$ is the set of ground actions*[1] *in Π.*
- *$\mathcal{O}R$ is a set of ordering constraints ($\prec$) over $\mathcal{O}$*
- *$\mathcal{C}L$ is a set of causal links over $\mathcal{O}$. A causal link is of the form (a_i, p, a_j), and denotes that the precondition p of action a_j will be supported by an* add *effect of action a_i.*
- *$\mathcal{O}C$ is the set of* open conditions *of Π. Let $a_i \in \mathcal{O}$; if $\exists p \in pre(a_i) \wedge \nexists a_j \in \mathcal{O}/(a_j, p, a_i) \subseteq CL$, then p is said to be an open condition.*
- *$\mathcal{U}L$ is the set of* unsafe causal links *of Π, also called the* threats*. Let $(a_i, p, a_j) \subseteq \mathcal{C}L$; (a_i, p, a_j) is unsafe if there exists an action $a_k \in \mathcal{O}$ such that $p \in del(a_k)$ and $\mathcal{O}R \cup \{a_i \prec a_k \prec a_j\}$ is consistent.*

[1] Partial-order planners are capable of handling partially instantiated action instances and hence, the definition of a partial order plan typically includes a set of equality constraints on free variables in $\mathcal{O}$ (43). We will, however, restrict our attention to ground action instances without any loss of generality for our purposes.

Given a planning task $\mathcal{T} = \langle \mathcal{O}, \mathcal{I}, \mathsf{G} \rangle$, a POP algorithm starts with an empty partial plan and keeps refining it until a solution plan is found. The initial empty plan $\Pi_0 = \langle \mathcal{A}P, \mathcal{O}R, \mathcal{C}L, \mathcal{O}C, \mathcal{U}L \rangle$ contains only two dummy actions $\mathcal{A}P = \{a_0, a_f\}$, the *start* action a_0, and the *finish* action a_f, where $pre(a_f) = \mathsf{G}$, $add(a_0) = \mathcal{I}$, $\{a_0 \prec a_f\} \subseteq \mathcal{O}R$, $\mathcal{C}L = \emptyset$, $\mathcal{O}C = \mathsf{G}$ and $\mathcal{U}L = \emptyset$. The empty plan has no causal links or threats, but, has open condition corresponding to the preconditions of a_f (the top-level goals G). A refinement step in a POP algorithm involves two things; first, selecting a flaw (an open condition or a threat) in a partial plan Π, and then selecting a resolver for the flaw. The different ways of solving a flaw are:

- Supporting an open condition with an *action step*. If p is an open condition, an action a needs to be selected that achieves p. a can be a new action from $\mathcal{O}$, or any action that already exists in $\mathcal{A}P$. Solving an open condition involves adding a causal link to Π to record that p is achieved by the chosen action step.
- Solving a threat with an *ordering constraint*. When the flaw chosen is an unsafe causal link (a_i, p, a_j) that is threatened by an action a_k, it can be repaired either by adding the ordering constraint $a_k \prec a_i$, or the constraint $a_j \prec a_k$, into $\mathcal{O}R$. This solving method involves reordering the action steps in Π.

Definition 7. *A plan $\Pi = \langle \mathcal{A}P, \mathcal{O}R, \mathcal{C}L, \mathcal{O}C, \mathcal{U}L \rangle$ is **complete** if it has no open conditions ($\mathcal{O}C = \emptyset$).*

Definition 8. *A plan $\Pi = \langle \mathcal{A}P, \mathcal{O}R, \mathcal{C}L, \mathcal{O}C, \mathcal{U}L \rangle$ is **conflict-free** if it has no unsafe causal links ($\mathcal{U}L = \emptyset$).*

Definition 9. *A plan $\Pi = \langle \mathcal{A}P, \mathcal{O}R, \mathcal{C}L, \mathcal{O}C, \mathcal{U}L \rangle$ is a **solution** if it is complete and conflict-free.*

2.2.3 Argumentation in POP

The task of the agents in classical planning is to be able to set goals and achieve them, i.e. finding a causal chain of actions that, when applied in the initial state, it achieves the

desired (sub)goals. In this sense, the set $pre(a)$ of a planning action a is interpreted as a set of *achievable* preconditions. However, actions can also have preconditions whose predicates are not affected by any of the actions available to the planning agent. Instead, the predicate's truth value is the result of a derivation obtained by forward chaining inference rules. More concretely, in our framework, the agent is equipped with a set of planning actions, $\mathcal{O}$, and a knowledge base $\mathcal{K} = (\Psi, \Delta)$ where:

- Ψ is a consistent set of facts. Initially, $\Psi = \mathcal{I}$, and this set will be updated accordingly with the add and del effects of the applicable actions.
- Δ is a set of *defeasible* rules that will be used to derive plausible information, tentative conclusions that might be withdrawn with new pieces of information.

In conclusion, a planning action a is a tuple $a = (pre(a), \mathsf{add}(a), del(a))$, where the set $pre(a)$ is divided into two subsets:

- $pre\_ach(a)$ denotes the set of *achievable* preconditions of the action a. The semantics is the same as in classical planning; an achievable precondition p of an action a is supported if it exists a set of actions from $\mathcal{O}$ that achieves the fact, and p **holds** in the state in which a will be applied, i.e. p is not deleted by any action before it holds in the state.
- $pre\_der(a)$ denotes the set of *derivable* preconditions of the action a, the set of preconditions that can be solved via a defeasible derivation. More particularly, the semantics is that a derivable precondition p of an action a is supported if there exists an argument $\langle A, p\rangle$ such that the root of a the tree $\mathcal{T}^*_{\langle \mathcal{A}, p\rangle}$ is marked undefeated, i.e. p **is warranted** in the state in which a will be applied.

Achievable preconditions are supported in a partial-order plan through action steps (see section 2.2.2.2). On the other hand, derivable preconditions are supported through argument steps as proposed in the argumentation-based formalism presented in (23). Hence,

we define a POP paradigm in combination with the argumentation formalism described in section 2.2.2.1, and we analyze the interplay of arguments and actions when constructing plans using POP techniques.

Definition 10. *Let $\mathcal{K} = (\Psi, \Delta)$ be the knowledge base of an agent; and let $\langle \mathcal{A}, p \rangle$, $\mathcal{A} \subseteq \Delta$, an argument that supports a derivable literal p. The set $facts(\mathcal{A})$ contains the facts that appear in the bodies of the rules in $\mathcal{A}$.*

In a partial-order plan Π, when an argument $\langle \mathcal{A}, p \rangle$ is used to support a derivable precondition p of an action a_i, Π will contain a new element, a ***support link*** of the form $(\mathcal{A}, p, a_i)$. This refinement step for solving a derivable precondition of an action is called *argument step* (23). Like causal links, support links are used to support a derivable precondition with the conclusion of an argument. Assuming an argument step $\mathcal{A}1 = \langle \mathcal{A}, p \rangle$, we can interpret that $add(\mathcal{A}1) = \{p\}$, and $pre(\mathcal{A}1) = facts(\mathcal{A}1)$. As can be observed, the introduction of argument steps does not imply any changes in the POP algorithm.

Under this new perspective, we reformulate the definition 6 as follows: A **partial-order plan** is a tuple $\Pi = \langle \mathcal{A}P \cup \mathcal{AR}, \mathcal{O}R, \mathcal{C}L \cup \mathcal{S}L, \mathcal{O}C \cup \mathcal{D}P, \mathcal{U}L \rangle$, where $\mathcal{A}P$, $\mathcal{O}R$, $\mathcal{C}L$, $\mathcal{O}C$ and $\mathcal{U}L$ have the usual meaning, $\mathcal{AR}$ is the set of argument steps included in Π, $\mathcal{S}L$ is the set of support links, and $\mathcal{D}P$ is the set of pending derivable preconditions of the actions in Π. Note that the facts of an argument step are the achievable preconditions of the argument and as such they are included as open conditions in the set $\mathcal{O}C$.

Unlike the approach presented in (23), DefPlanner is a defeasible argumentation-based planner in which many different agents with different opinions argue with each other on the warranty of a given argument. During the plan construction, at the time of solving a derivable precondition p, DefPlanner invokes a procedure and agents initiate a discussion in order to check whether p can be warranted or not. This procedure builds a dialectical tree for each supporting argument of p and finally returns whether p is defeated or undefeated. This *multi-agent discussion* is explained in detail in next section. Hence, in the case of DefPlanner, argument steps are only inserted in a partial-order plan as long

as it has been proven the argument is undefeated. This contrasts with other approaches in which each supporting argument gives rise to a different alternative in the POP algorithm, and discussions on the warranty of a given argument take place in case a counter-argument is introduced in the plan. In conclusion, DefPlanner only inserts provably undefeated arguments in a plan, and, consequently, no threats involving two argument steps may appear in our approach. Let $\langle \mathcal{A}1, p \rangle$ be an argument step inserted in a plan Π; if argument $\langle \mathcal{A}2, q \rangle$ is later inserted in Π then DefPlanner guarantees $\mathcal{A}2$ is not a counter-argument of $\mathcal{A}1$ and viceversa.

Additionally, in this first approach of DefPlanner, we assume a piece of information can not be both derived and achieved. That is, a proposition p is either defeasibly derived through a dialectical tree by using the rules in Δ, or achieved through a course of actions in $\mathcal{O}$. Thus, the predicates of defeasible information are never affected by the available planning actions $\mathcal{O}$ and, consequently, no action-argument threats exist. In section 2.2.6, we elaborate on this issue for future versions of DefPlanner.

2.2.4 Defeasible argumentation in a multi-agent system

DefPlanner implements a Multi-Agent System (MAS) (figure 2.1) to assist during the construction plan. Agents can adopt one of the four different roles specified in this MAS:

- *Client role*: The user is represented by an agent playing this role, which is in charge of requesting a plan for a given set of goals.
- *POP role*: The agent playing this role, that is, the planner takes as input the set of goals and returns a solution plan that satisfies the client goals. There is only one agent playing the POP role per MAS.
- *Argumentative role*: An agent ag_i which plays this role is associated with a set of defeasible rules representing the tentative information of the agent about the environment (Δ_i). The task of each argumentative agent ag_i is to participate as far

as possible in the multi-agent discussions for warranting a given literal. Each agent has an associated utility function[1] that is used to maximize its benefits.

- *Mediator role*: The agent which plays this role (only one per MAS) is in charge of managing the multi-agent argumentation process.

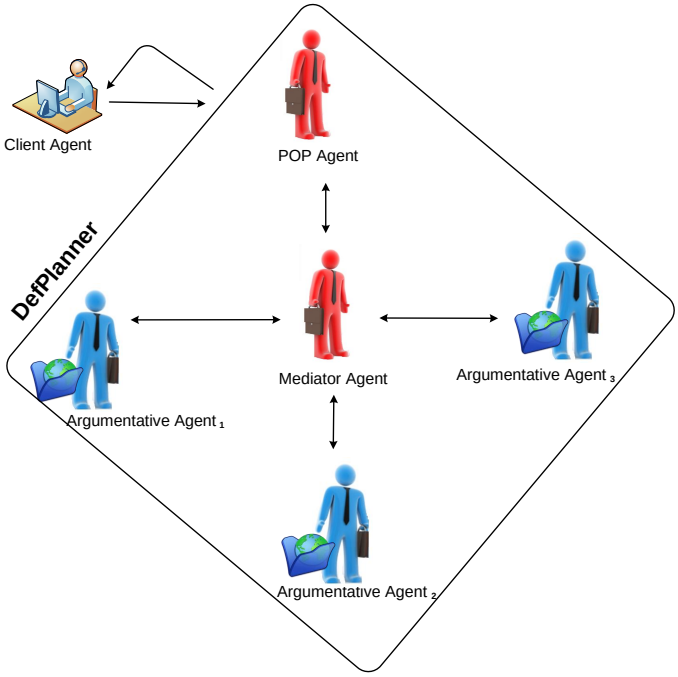

Figure 2.1: An overview of DefPlanner.

A MAS, as defined in this paper, is formed by a POP agent which reasons about which action step (for solving an open condition), or ordering constraint (for solving a threat) should be chosen in the next iteration of the POP algorithm; a group of non-self-interested argumentative agents, which join together to reason about the argument step

[1]For instance, in terms of less cost, time, resources or increased safety could be expressed their utility functions.

that should be chosen to satisfy/warrant a derivable precondition; and, a mediator agent, which coordinates the multi-agent argumentation process for warranting a literal.

2.2.4.1 DefPlanner Algorithm

The POP agent implements an extension of the traditional POP algorithm by considering the introduction of argument steps, and corresponding support links, to resolve a defeasible precondition (Algorithm 1). The three non-determinist **choose** statements state that the algorithm has to make a choice among different alternatives (selecting the next partial-plan to work on, selecting a pending derivable precondition in the partial plan, or selecting the next open condition/threat to study). Typically, the selected choice will be the result of the application of a specific heuristic (36). The $multi\_argumentation$ function encodes the defeasible argumentation multi-agent process, which will be explained in detail in the next subsection.

The traditional POP algorithm works as follows: starting with the initial empty plan Π_0 (step 1 in Algorithm 1), it works through the application of successive refinement steps at each iteration. First, it chooses a partial-order plan from the list of candidates (step 3 in Algorithm 1), and then it applies a refinement step that involves selecting a flaw (threat or open condition) in the partial-order plan (step 11 in Algorithm 1).

In contrast with the traditional POP algorithm, the new algorithm considers argument steps, besides action steps, to support unsatisfied derivable preconditions. The POP agent takes an argument step as the support from the defeasible argumentation multi-agent process (section 2.2.4.2) to derive a defeasible precondition. If no argument steps can be constructed to support a derivable precondition, then it prunes[1] the selected plan Π from $Plan\_List$. Note that, unlike the achievable preconditions, the algorithm does not branch for each different argument step that supports a derivable precondition. As it will be explained later, in case of more than one undefeated argument step for a given defeasible

[1] i.e. the plan is discarded and the search process does not continue exploring through this plan.

precondition, the voting phase will select the best argument step according to the preference criterion of generalized specificity (see section 2.2.2.1).

The process ends when both $subgoal\_list\_1$ and $subgoal\_list\_2$ are empty, in whose case Π is a solution plan, or when $Plan\_List$ is empty, in whose case there is not a solution plan.

1: $Plan\_list := \Pi_0$
2: **repeat**
3: **choose** $\Pi \in Plan\_list$
$subgoal\_list\_1 := \mathcal{D}P(\Pi)$
$subgoal\_list\_2 := \mathcal{O}C(\Pi) \bigcup \mathcal{U}L(\Pi)$
4: **if** $(subgoal\_list\_1 \bigcup subgoal\_list\_2 = \emptyset)$ **then**
5: return Π {Plan solution}
6: **else if** $subgoal\_list\_1 \neq \emptyset$ **then**
7: **choose** $\Phi \in subgoal\_list_1$
$\Pi_a := multi\_argumentation(\Phi)$
8: **if** $\Pi_a \neq NIL$ **then**
9: then $Plan\_list := Plan\_list \bigcup \Pi_a$
10: **else**
11: **choose** $\Phi \in subgoal\_list_2$
$Relevant := \{\Pi_r\}, \forall\ \Pi_r$ that resolves Φ {*Each r is a choice (partial-order planning) to solve* Φ}
12: **if** $Relevant \neq \emptyset$ **then**
13: $Plan\_list := Plan\_list \bigcup Relevant$
14: **until** $Plan\_list = \emptyset$
15: return fail {*Not exists plan*}

Algorithm 1: Outline of the DefPlanner algorithm

2.2.4.2 Defeasible Argumentation Multi-Agent Process

The objective of this process is to have multiple agents reasoning (discussing) about the warrant for a particular derivable precondition p requested by the POP agent. The output

of the process will be an argument step, if it exists an undefeated argument structure for p; otherwise, the procedure will return NIL (step 8 in Algorithm 1) thus indicating there is no refinement plan that supports the defeasible precondition p.

In what follows, we will consider the notions defined in the section 2.2.2.1, such as argument structure, disagreement, argumentation line, etc. Unlike single-agent contexts, in our multi-agent framework arguments and counter-arguments will be proposed by different argumentative agents in the MAS.

DefPlanner divides the reasoning process into three phases: the Dialogue Phase, in which arguments and counter arguments are proposed, the Evaluation Phase, in which each argument proposal to derive p is marked as defeated or undefeated, and the Voting Phase, in which a voting is applied - in case of more than one undefeated argument structure- to choose the best undefeated proposal for p according to the preference criterion.

Dialogue Phase. Both the argumentative agents and the mediator agent are involved in this phase. The argumentative agents of the MAS provide two functionalities: (I) propose an initial argument structure to support a derivable precondition p, which will be the root of a dialectical tree, and (II) propose a counterargument to the argument articulated by another agent in the argumentation line. We assume that argumentative agents are ordered according to their indexes: 1, 2, ..., n. The proposed model follows a rotating shift approach[1], in which an argumentative agent can only participate during its turn. The mediator agent is in charge of adding the proposed arguments to the appropriate dialectical tree or creating a new dialectical tree in case of a new initial argument structure.

Let $\langle \mathcal{X}^i, h \rangle$ be an argument structure where $\mathcal{X}$ is the argument support, i denotes the argumentative proposer agent, and h is the conclusion supported by the argument. Extending the definition 4 (section 2.2.2.1), an argumentation line in DefPlanner, $\Lambda =$

[1]The shift approach allows to treat uniformly each agent.

$[\langle \mathcal{X}^i, h_0\rangle, \langle \mathcal{Y}^j, h_1\rangle \ldots, \langle, \mathcal{Z}^i, h_n\rangle]$, is a sequence of argument structures from different argumentative agents such that two consecutive argument structures cannot be proposed by the same agent; i.e. $\langle \mathcal{X}^i, h_0\rangle$, $\langle \mathcal{Y}^j, h_1\rangle$, and $i \neq j$. Thereby, DefPlanner does not allow agents giving counterarguments to their own arguments, and this is achieved by ensuring that the agent's local belief base (Δ_i) is consistent with respect to the global belief base (Ψ). In this first version of DefPlanner, at the turn of an argumentative agent, it has to articulate all the arguments for a given derivable precondition, or all its counterarguments for a given argument, so an agent can jump-shift the turn only if it lacks sufficient information to make a new proposal. However, in future versions, we will consider to model other different kinds of argumentation strategies.

Specifically, the aim of this phase is to provide reasons that support a derivable precondition $p \in pre\_der(a)$. A new argument $\langle \mathcal{X}^i, p\rangle$ represents the root of a dialectical tree $\mathcal{T}_{\langle \mathcal{X}, p\rangle}$. In order to determine whether $\langle \mathcal{X}^i, p\rangle$ is an undefeated argument or not in the next phase, agents alternatively propose a counter-argument as a defeater to any of the leaf nodes of the dialectical tree $\mathcal{T}_{\langle \mathcal{X}, p\rangle}$. According to (24), a counter-argument $\langle \mathcal{Y}^j, h_2\rangle$ to the argument $\langle \mathcal{X}^i, h_1\rangle$ can be a *direct* attack to the conclusion, that is h_2 and h_1 are contradictory literals, or can be an indirect attack by arguing an inner point h of $\langle \mathcal{X}^j, h_1\rangle$. Since counter-arguments are arguments too, there may exist defeaters for them, and so on, thus giving rise to the argumentation lines of $\mathcal{T}_{\langle \mathcal{X}, p\rangle}$.

Evaluation Phase. At this phase, the aim is to decide whether a dialectical tree of a defeasible precondition p is marked as undefeated or defeated. More specifically, the mediator agent performs a bottom-up-analysis for each dialectical tree $\mathcal{T}_{\langle \mathcal{X}, p\rangle}$ developed in the above phase, obtaining a set of marked dialectical trees $\bigcup_{0}^{n} \mathcal{T}^*_{\langle \mathcal{X}, p\rangle}$, where n is the total number of dialectical trees for p. Nodes will be recursively marked as D (*defeated*) or U (*undefeated*) like the minimax tree used in Artificial Intelligence for game trees. At the end of this process, each root argument $\langle \mathcal{X}^i, p\rangle$ will be marked as defeated or undefeated

(Definition 5). In DefPlanner, a derivable precondition p is warranted if it has at least a root argument that satisfies p, and the corresponding dialectical tree is marked as U (*undefeated*).

Voting Phase. If the derivable precondition p has more than one undefeated argument, we must choose one of them as the support for p in a partial-order plan Π. In this phase, agents vote the most preferable undefeated argument according to their own utility function. The undefeated argument structure with the highest number of votes will be the selected argument step to be included in Π. In case of tie-breaking, the mediator agent makes the final decision. So, the voting idea is that each agent votes according to their different partial plans. For instance, the next utility function could be adopted: generalized specificity (44), a function that favors two aspects in an argument to derive a derivable precondition: it prefers (1) a more precise argument (i.e., with greater information content) or (2) a more concise argument (i.e., with less use of rules). So, the undefeated arguments with greater information and less rules would be preferred.

The following section illustrates the application of this protocol to an example scenario in order to obtain a solution plan for a planning task.

2.2.5 Example of application

Figure 2.21 shows the planning scenario where we will put our argumentation-based model to work. There are two different locations in this scenario $l1$ and $l2$. As can be seen in the figure, there are three different connections between $l1$ and $l2$: via truck, train or plane, and so the client agent can reach $l2$ by using any of these three transport means. The client agent, the truck, the train and the plane are initially located at $l1$. The goal of the problem is to have the client agent in $l2$. Following, we present the objects defined in this problem:

- $l1$, $l2$, ca - location 1, location 2, and the client agent

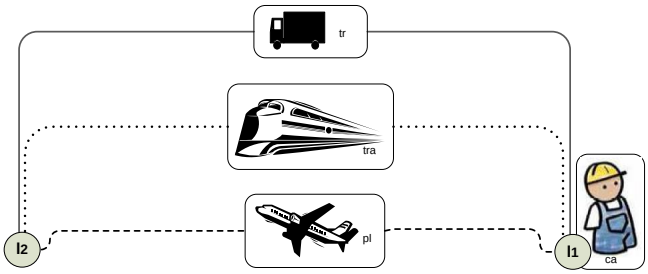

Figure 2.2: Scenario of the application example

- tr, tra, pl - a truck, a train, a plane,
- r, tl, al, ae - a road, a railway, an airline company, the airline experts,
- tv, in, - television news, internet news
- bw, sn, wg, - bad weather, snow, wind gusts
- br, vi, ll, esf - bad railroad, adequate visibility, landslides, electrical supply failure
- rm, va, ds - airplane engines work well, volcano ash cloud hits airline, dangerous situation
- h, $j6$, tj - holidays, June 6, and traffic jam.

The actions the client agent can perform are the following ones:

- Mp$(?j, ?k)$: moving plane pl from location j to k. It must exist an airline company to travel from j to k, and absence of dangerous situations to assure safety. Moving a plane takes 3 time units.

$$\mho = \left\{ \begin{array}{l} \{(at\ tr\ ?k),\ \sim(at\ tr\ ?j),\ (at\ ca\ ?k),\ \sim(at\ ca\ ?j)\} \xleftarrow{mFt} \\ \quad \{(link\ r\ ?j\ ?k),\ (at\ tr\ ?j),\ (at\ ca\ ?j),\ \sim(tj)\} \\ \\ \{(at\ tr\ ?k),\ \sim(at\ tr\ ?j),\ (at\ ca\ ?k),\ \sim(at\ ca\ ?j)\} \xleftarrow{mSt} \\ \quad \{(link\ r\ ?j\ ?k),\ (at\ tr\ ?j),\ (at\ ca\ ?j)\} \\ \\ \{(at\ tra\ ?k),\ \sim(at\ tra\ ?j),\ (at\ ca\ ?k),\ \sim(at\ ca\ ?j)\} \xleftarrow{mT} \\ \quad \{(link\ tl\ ?j\ ?k),\ (at\ tra\ ?j),\ (at\ ca\ ?j),\ \sim(br),\ (v)\} \\ \\ \{(at\ pl\ ?k),\ \sim(at\ pl\ ?j),\ (at\ ca\ ?k),\ \sim(at\ ca\ ?j)\} \xleftarrow{mP} \\ \quad \{(link\ al\ ?j\ ?k),\ (at\ pl\ ?j),\ (at\ ca\ ?j),\ \sim(ds)\} \end{array} \right\}$$

- fMt$(?j, ?k)$: fast-moving truck tr from location j to k. It must exist a road from j to k, and assure there is no traffic jam between j and k. This action takes 8 time units.

- sMt$(?j, ?k)$: slow-moving truck tr from location j to k. It must exist a road from j to k. This action takes 20 time units.

- Mt$(?j, ?k)$: moving train tra from location j to k. There must exist a railway from j to k, and no bad railroad conditions to assure an adequate visibility. This action takes 10 time units.

Our multi-agent system consists of the POP agent, the mediator agent and three argumentative agents, Bob, Joe and Ann. Agents have different knowledge and two pieces of information from different agents can appear to be contradictory. Let's assume that each argumentative agent is a travel agency, that Joe uses TV as a source of information, but Ann prefers Internet to keep up to date. The goal ($\mathcal{G}$) is to have the agent ca at position

$l2$, $(at\ ca\ l2)$. The global belief base (Ψ), the local belief bases (Δ_{Bob}, Δ_{Joe}, Δ_{Ann}), and the action base ($\mathcal{O}$) are detailed as follows:

$$\Psi = \left\{ \begin{array}{c} (have\ in);\ (have\ tv);\ (have\ vi);\ (have\ va); \\ (have\ wg);\ (today\ j6);\ (have\ ae);\ (at\ ca\ l1); \\ (at\ tr\ l1);\ (at\ pl\ l1);\ (at\ tra\ l1); \\ (link\ l1\ l2\ r);\ (link\ l1\ l2\ tl);\ (link\ l1\ l2\ al); \end{array} \right\}$$

$$\Delta_{Bob} = \left\{ \begin{array}{c} br \prec\!\!\!- ll;\ ll \prec\!\!\!- wg;\ bw \prec\!\!\!- wg; \\ ds \prec\!\!\!- \{va, tv\}; \end{array} \right\}$$

$$\Delta_{Joe} = \left\{ \begin{array}{c} br \prec\!\!\!- esf;\ esf \prec\!\!\!- sn;\ br \prec\!\!\!- sn;\ \sim bw \prec\!\!\!- sn; \\ sn \prec\!\!\!- tv;\ tj \prec\!\!\!- h; \\ h \prec\!\!\!- j6;\ \sim ds \prec\!\!\!- rm;\ rm \prec\!\!\!- ae; \end{array} \right\}$$

$$\Delta_{Ann} = \left\{ \begin{array}{c} \sim bw \prec\!\!\!- h;\ h \prec\!\!\!- j6;\ \sim ll \prec\!\!\!- \sim bw; \\ \sim br \prec\!\!\!- \sim bw; \sim bw \prec\!\!\!- in;\ \sim sn \prec in;\ \sim rm \prec\!\!\!- va; \end{array} \right\}$$

For the sake of simplicity, Bob, Joe and Ann have the same utility function. Specifically, we consider the same comparison criterion among defeaters arguments (section 2.2.2.1), as an utility function that returns the best undefeated argument. In what follows, we explain how DefPlanner works to obtain a complete plan Π that satisfies the goal $\mathcal{G}$.

2.2.5.1 Searching for a Solution Plan

Step 1. The planning process starts with the empty plan Π_0 (leftmost plan in Figure 2.5). For solving the precondition $\Phi = (at\ ca\ l2)$, the POP agent has four different action choices $\{mP(l1,\ l2),\ mFt(tr,\ l1,\ l2),\ mT(l1,\ l2),\ mSt(tr,\ l1,\ l2)\}$, so four new partial-order plans $\{\Pi_{0.1}, \Pi_{0.2}, \Pi_{0.3}, \Pi_{0.4}\}$ are added to $Plan\_List$ (see Figure 2.6).

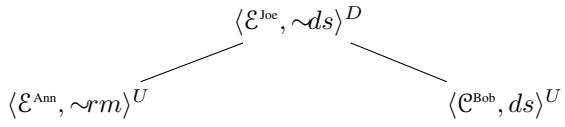

Figure 2.3: Marked dialectical tree for the derivable precondition $\sim ds$ at step 2 of the plan solution process

Step 2. Let's assume the POP selects the plan $\Pi_{0.1}$ because it is the plan that takes fewer time units. Then we have $\mathcal{A}P(\Pi_{0.1}) = \{mP(l1,\ l2)\}$. The action $mP(l1,\ l2)$ has a derivable precondition $p = \sim ds$ meaning that the plane can only fly if it is assured that no dangerous situation is expected during the flight. The POP agent invokes the mediator agent that calls the $multi\_argumentation$ function, and it proposes a new dialogue phase to check whether p is warranted or not.

Joe takes the first shift, and puts forward the initial argument $\langle \mathcal{E}^{\text{Joe}}, \sim ds \rangle$ with $\mathcal{E}^{\text{Joe}} = \{\sim ds \prec\!\!\!- rm;\ rm \prec\!\!\!- ae\}$, indicating that the airline experts assert the airplane engines work well and that there will be no dangerous situation. When counterarguments to this argument are requested, Ann responds[1] with $\langle \mathcal{E}^{\text{Ann}}, \sim rm \rangle$ with $\mathcal{E}^{\text{Ann}} = \{\sim rm \prec\!\!\!- va\}$, and Bob responds $\langle \mathcal{C}^{\text{Bob}}, ds \rangle$ with $\mathcal{C}^{\text{Bob}} = \{ds \prec\!\!\!- \{va, tv\}\}$. Nobody has more information to argue against, so the process ends here. Figure 2.3 shows the argument $\langle \mathcal{E}^{\text{Joe}}, \sim ds \rangle$ is marked as defeated, and, consequently, $\sim ds$ is not warranted. The $multi\_argumentation$ function returns $\Pi_{0.1.1} = NIL$ because $\sim ds$ is not warranted. Thereby, $\Pi_{0.1}$ is discarded from the $Plan\_list$.

Step 3. Let's assume the next plan to be selected is $\Pi_{0.2}$ (see Figure 2.6), where = $\mathcal{A}P(\Pi_{0.2}) = \{mFt(tr\_1,\ l1,\ l2)\}$. The action $mFt(tr\_1,\ l1,\ l2)$ has a derivable precondition $p = \sim tj$, indicating that there should not be traffic jam for fast-moving truck. The POP agent selects $\Pi_{0.2}$ because it is the second plan with fewer time units. The POP

[1] An argumentative agent responds if it is at its turn.

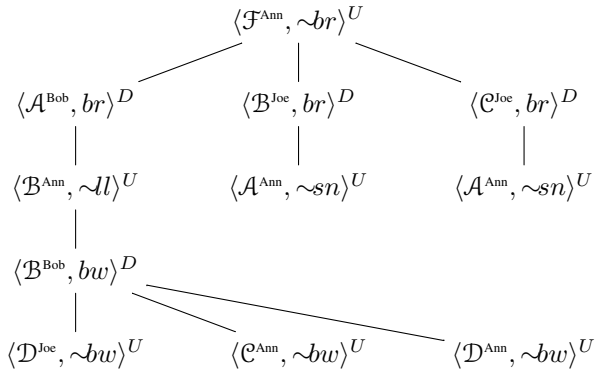

Figure 2.4: Marked dialectical tree for the derivable precondition $\sim br$ at step 2 of the plan solution process

agent invokes the mediator agent that calls the $multi\_argumentation$ function, and it proposes a new dialogue phase to check whether $\sim tj$ is warranted or not.

Bob and Ann have not traffic jam information, and Joe knows there is traffic jam because today is June 6, and $\{tj \prec\!\!- h;\ h \prec\!\!- j6\}$. Therefore, nobody can support $\sim tj$, and, $multi\_argumentation$ function returns $\Pi_{0.2.1} = NIL$. Thereby, $\Pi_{0.2}$ is discarded from $Plan\_list$.

Step 4. Assuming the next selected plan is $\Pi_{0.3}$ (Figure 2.6) where $\mathcal{A}P(\Pi_{0.3}) = \{mT(l1,\ l2)\}$, the action $mT(l1,\ l2)$ has a derivable preconditions which indicates the railroad must not be in bad conditions; then, $\Phi = \sim br$.

Ann takes the first shift, and puts forward the initial argument $\langle \mathcal{F}^{\text{Ann}}, \sim br\rangle$ with $\mathcal{F}^{\text{Ann}} = \{\sim br \prec\!\!- \sim bw; \sim bw \prec\!\!- in\}$, i.e., internet news say that bad weather is not expected, and, therefore, the railroad will not be in bad conditions. Next, Bob takes the shift and responds directly attacking $\sim br$ with $\langle \mathcal{A}^{\text{Bob}}, br\rangle$, where $\mathcal{A}^{\text{Bob}} = \{br \prec\!\!- ll;\ ll \prec\!\!- wg\}$, meaning that wind gusts are expected according to the information in the initial state, and because of

that landslides may occur. If landslides happen to occur, then it is likely the case to have the railroad in bad conditions.

Joe takes the shift, and responds to $\sim br$ with $\langle \mathcal{B}^{\text{Joe}}, br\rangle$ with $\mathcal{B}^{\text{Joe}} = \{br \multimap esf;\ esf \multimap sn;\ sn \multimap tv\}$, and, $\langle \mathcal{C}^{\text{Joe}}, br\rangle$ with $\mathcal{C}^{\text{Joe}} = \{br \multimap sn;\ sn \multimap tv\}$. That is, according to Joe's information, television news report it will snow, and so the railroad is likely to be in bad conditions as well as having a electrical supply failure, which causes to have the railroad in bad conditions.

When counterarguments to $\langle \mathcal{B}^{\text{Joe}}, br\rangle$ and $\langle \mathcal{C}^{\text{Joe}}, br\rangle$ are requested, Ann responds with $\langle \mathcal{A}^{\text{Ann}}, \sim sn\rangle$, where $\mathcal{A}^{\text{Ann}} = \{\sim sn \prec in\}$. When asked to counter-argue $\langle \mathcal{A}^{\text{Bob}}, br\rangle$, Ann responds with $\langle \mathcal{B}^{\text{Ann}}, \sim ll\rangle$ where $\mathcal{B}^{\text{Ann}} = \{\sim ll \multimap \sim bw; \sim bw \multimap in\}$. According to Ann's information, internet reports that no bad weather is expected and so there is no chance to find landslides.

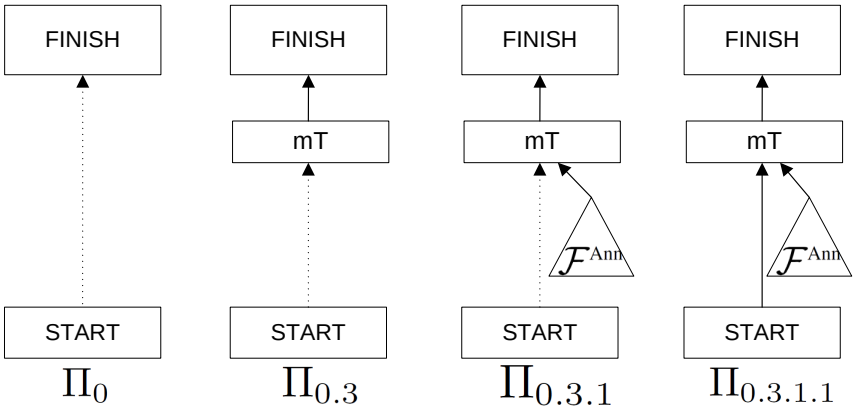

Figure 2.5: Different partial plans for the example scenario

In turn, when asked to counter-argue $\langle \mathcal{B}^{\text{Ann}}, \sim ll\rangle$, Bob takes the shift, and responds

$\langle \mathcal{B}^{\text{Bob}}, bw \rangle$ with $\mathcal{B}^{\text{Bob}} = \{bw \multimap wg\}$. In turn, Joe responds $\langle \mathcal{D}^{\text{Joe}}, \sim bw \rangle$ with $\mathcal{D}^{\text{Joe}} = \{\sim bw \multimap sn;\ sn \multimap tv\}$, and, Ann responds $\langle \mathcal{C}^{\text{Ann}}, \sim bw \rangle$ with $\mathcal{C}^{\text{Ann}} = \{\sim bw \multimap h;\ h \multimap j6\}$, and $\langle \mathcal{D}^{\text{Ann}}, \sim bw \rangle$ with $\mathcal{D}^{\text{Ann}} = \{\sim bw \multimap in\}$.

Figure 2.4 shows that the argument $\langle \mathcal{F}^{\text{Ann}}, \sim br \rangle$ is marked as undefeated, and, consequently, the derivable precondition $\sim br$ is warranted. The $multi\_argumentation$ function returns $\Pi_{0.3.1}$, an extension of $\Pi_{0.3}$ with $\mathcal{F}^{\text{Ann}}$ and $CL(\mathcal{F}^{\text{Ann}})$.

Step 5. Assuming the plan selected next is $\Pi_{0.3.1}$ (because it has less duration than $\Pi_{0.4}$), the POP agent extends $\Pi_{0.3.1}$ to $\Pi_{0.3.1.1}$, adding a causal link between $facts(\mathcal{F}^{\text{Ann}})$ and the initial state Ψ (Figure 2.6). $\Pi_{0.3.1.1}$ is a solution plan that satisfies the goal $\mathcal{G}$ (Figure 2.5).

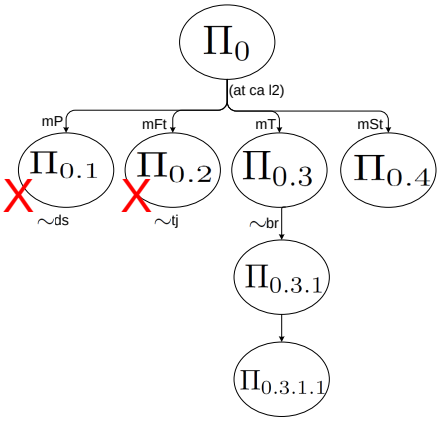

Figure 2.6: Search in the space of partial-order plans for the example scenario

2.2.6 Conclusions and related work

In this paper, we have presented DefPlanner, a defeasible argumentation-based planner that allows multiple agents with partial and contradictory knowledge articulate reasons for and against the precondition of a planning action. Along the paper, we have introduced the necessary modifications to include a defeasible reasoning into a POP algorithm. This new and enriched planner opens up many possibilities to be applied to a multi-agent planning context.

DefPlanner builds on the approximation of Garcia et al (23), and extends their work by incorporating multiple agents at the time of deciding which literals (conditions of a planning action or derivable preconditions) are warranted. Our work is also related to conformant planning (47), an approach to deal with planning with incomplete information in which the purpose is to generate plans given uncertainty about the initial state and action effects, and without any sensing capabilities during plan execution. However, unlike conformant planning, our approach is a powerful planning mechanism for reasoning about contradictory information coming from different sources or agents. In this sense, in the literature of classical planning we can hardly find approaches to deal with contradictory information because, among other reasons, there are very few attempts to extend planning to a multiagent environment, being a notably exception the work of Brenner and Nebel (48). Hence, DefPlanner is a novel approach regarding the consideration of incomplete and contradictory information of multiple reasoning entities, i.e. agents.

As for future work, we are interested in extending the argumentation process to achievable preconditions; that is, a new approach towards the integration of reasoning about action steps (practical reasoning) and reasoning about argument steps (defeasible reasoning). Particularly, our next immediate step is to endow agents with planning capabilities, rather than just limiting agents to perform defeasible reasoning and discuss the warranty of literals, and thus come up with a defeasible multiagent planning approach. In this

context, we will also study the choice of having non-cooperative agents in the MAS.

Acknowledgements

We would like to thank three anonymous reviewers for helpful comments that have helped to improve this work. This work is supported by FPU grant reference AP2009-1896 awarded to Sergio Pajares-Ferrando, TIN2008-04446, TIN2008-06701-C03-03 and PROMETEO/2008/051 projects of the Spanish government and CONSO-LIDER INGENIO 2010 under grant CSD2007-00022.

2.3 Selected Paper 2: An architecture for Defeasible-Reasoning-based Cooperative Distributed Planning (CoopIS 2011)

Abstract. *Cooperation plays a fundamental role in distributed planning, in which a team of distributed intelligent agents with diverse preferences, abilities and beliefs must cooperate during the planning process to achieve a set of common goals. This paper presents a **M**ulti**A**gent **P**lanning and **A**rgumentation (*MAPA*) architecture based on a multiagent partial order planning paradigm using argumentation for communicating agents. Agents use an argumentation-based defeasible reasoning to support their own beliefs and refute the beliefs of the others according to their knowledge. In* MAPA*, actions and arguments may be proposed by different agents to enforce some goal, if their conditions are known to apply and arguments are not defeated by other arguments applying. In order to plan for these goals, agents start a stepwise dialogue consisting of exchanges of plan proposals to satisfy this open goal, and they evaluate each plan proposal according to the arguments put forward for or against it. After this, an agreement must be reached in order to select the next plan to be refined.*

2.3.1 Introduction

A Cooperative Information System (CIS) is a large scale information system that interconnects various systems of different and autonomous organizations, geographically distributed and sharing common objectives (49). With the emergence of new technologies in computing, such as SaaS, cloud computing, Service Oriented Computing, mash-ups, Web Services, Semantic Web, Knowledge Grid, and other approaches, it is becoming increasingly natural to deal with Agent-based computing or **MultiAgent Systems**(50). Agents, as distributed autonomous software entities, are required to engage in interactions, argue with one another, make agreements, and make proactive run-time decisions, individually

and collectively, while responding to changing circumstances. For this reason, agents are being advocated as a next-generation model for engineering complex distributed systems.

Planning is the art of building control algorithms that synthesize a course of action to achieve a desired set of goals of the information system. Unlike classical planning, in many real-world applications agents often have distributed contradictory information about the environment and their deductions are not always certain information, but *plausible*, since the conclusions can be withdrawn when new pieces of knowledge are posted by other agents. For this purpose, argumentation, which has recently become a very active research field in computer science (5, 37), can be viewed as a powerful tool for reasoning about inconsistent information through a rational interaction of arguments for and against some conclusion.

Defeasible Logic Programming (DeLP) (24) is a framework for reasoning about defeasible information (also known as defeasible reasoning), where tentative conclusions are obtained from uncertain or incomplete information, and conclusions might no longer be valid after new information becomes available. The work in (23) (see section 2.7.2) introduces a first approach known as DeLP-POP framework, to integrate DeLP in Partial Order Planning (POP) (43), and the work in (26) (see section 2.7.2) extends DeLP-POP framework to a multiagent environment. As an example on how defeasible reasoning is introduced in these frameworks, we can view an agent as a business person who needs to travel between London and Athens, and has to build a plan to get to Athens. One may think the first action to do is to buy a flight ticket through an airline web site. However, another agent who is aware of the latest news on the Internet, might think the business man will not be able to fly due to a strike announcement in London. Under these circumstances, the second agent will put forward an argument against the first one in order to ensure that the business man accomplishes his goal to get Athens.

The motivation for introducing distributed planning in a multi-agent environment is twofold. On one hand, a multi-agent system design can be beneficial in many domains,

particularly when a system is composed of multiple entities that are distributed functionally or spatially. On the other hand, distributed execution promotes the efficiency of parallel processing of actions, the robustness of the system to cope with complex planning problems and the simplicity of an incremental construction across a network of interconnected agents, thus avoiding the critical failures and resource limitations of centralized systems. In this paper, we present a **M**ulti**A**gent **P**lanning and **A**rgumentation (MAPA) architecture for cooperative distributed planning in a multiagent DeLP-POP framework, which extends and refines the preliminary work presented in (26). This paper is organized as follows: section 2 gives a short related work; section 3 describes a background; section 4 introduces the MAPA architecture; section 5 presents the planning protocol of the architecture; and section 6 shows an example of application to validate the MAPA architecture. Finally, we conclude and present some directions for future work.

2.3.2 Related Work

This subsection is devoted to study the most relevant related works found in the literature: multi-agent argumentation, cooperative distributed planning (without defeasible reasoning) and centralized planning. Some systems that build on argumentation apply theoretical reasoning for the generation and evaluation of arguments to build applications that deal with incomplete and contradictory information in dynamic domains. Some proposals in this line focus on planning tasks, or also called practical reasoning, i.e. reasoning about what actions are the best to be executed by an agent in a given situation. Dung's abstract system for argumentation (4) has been used for reasoning about conflicting plans and generating consistent sets of goals (51). Further extensions of these works present an explicit separation of the belief arguments and goal arguments and include methods for comparing arguments based on the value of goals and the cost of resources (5). The combination of defeasible reasoning and planning has been used in (52), in which the whole plan is viewed as an argument and then, defeasible reasoning about complete plans

is performed. Although the work in (52) combines defeasible reasoning and partial order planning, defeasible reasoning is not used in the same way as (23). In contrast, (23) uses arguments for warranting subgoals, and hence, defeasible reasoning is used in each step of the planning search process. In any case, none of these works apply to a multi-agent environment.

A proposal for dialogue-based centralized planning is that of (53), but no argumentation is made use of. The work in (29) presents a dialogue based on argumentation to reach agreements on plan proposals. Unlike our proposal, which focuses on an argumentative and stepwise construction of a plan, this latter work is aimed at handling the interdependencies between agents' plans. The work in (18) introduces a framework to build joint plans supported through the use of *argumentation schemes* as a mechanism of dialogue during the planning search. On the other hand, we can also find some systems that perform argumentation in multi-agent systems by using defeasible reasoning but are not particularly concerned with the task of planning (41).

2.3.3 Background

The key element of DeLP are defeasible rules (Head $-\prec$ Body), which are used to represent a deductive relation between pieces of knowledge that could be defeated once other piece of knowledge is considered. Specifically, arguments (combinations of defeasible rules and facts) for conflicting pieces of information are built, and then compared to decide which one prevails. For instance, a defeasible rule like "According to Internet news, an airport strike is expected", is denoted as "$strike -\prec news$". Note that, if it occurs in London, then it will disrupt the passengers' plans for flying between London and Athens.

The principle of least commitment in Partial Order Planning makes it one of the more open planning frameworks. This is evidenced by the fact that most existing architectures for integrating planning with execution, information gathering, and scheduling are

based on partial order planners. In (54), authors argue that POP-based frameworks offer a more promising approach for handling domains with durative actions, and temporal and resource constraints as compared to other planning approaches. In fact, most of the known implementations of planning systems capable of handling temporal and durative constraints (including IxTET (55), as well as NASA's RAX (56)) are based on the POP paradigm. Even for simple planning tasks, partial order planners offer a higher degree of execution flexibility. In contrast, none of the known state-space planners can find parallel plans efficiently (57), and planners such as Graphplan (58) only generate a very restricted types of parallel plans. For this reason, partial order planning remains attractive when compared to state-space planning.

An extension of POP with DeLP-style argumentation, denoted DeLP-POP framework, was introduced in (23), where both actions and arguments may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. Unlike actions, arguments will not only be introduced to intentionally support some step of a plan, but they will also be presented to defeat or defend other supporting arguments in the plan. When actions and arguments are combined in a partial order plan, new types of interferences or threats appear (23). These interferences need to be identified and resolved to obtain valid plans.

Finally, the work in (26) proposes a preliminary extension of the theoretical DeLP-POP framework to a multiagent environment. Specifically, it proposes a dialogue for argumentative plan search, by which agents exchange plan proposals and arguments for or against such proposals. Unlike (26), the MAPA architecture presented here solves the qualification problem, identifies new types of threats, and extends the agents' knowledge bases by including a set of agent-specific preferences. This allows us to extend and adapt the planning protocol of MAPA to a fully-automated argumentative dialogue between agents so as to reach agreements during the plan construction. Moreover, the MAPA

architecture promotes a more practical vision of the extension of DeLP-POP to a multi-agent environment.

2.3.4 Elements of the MAPA Architecture

In state-based planning, a plan Π is a linear sequence of actions, and thus before each action is added to the plan Π, we know which consistent state will hold. In contrast, MAPA architecture is based on POP[1], where a partial order plan Π is a set of actions whose execution ordering $\prec$ is only partially specified (thus encoding multiple linear plans).

The MAPA architecture works on a planning process distributed among several planning agents, which have an incomplete knowledge (i.e. the set of actions and arguments that an agent can propose can be different from other agents'), and have to devise a joint, non-linear plan which may be later executed by them. The following subsections expose (i) the agents' planning model and the notion of argument, (ii) the improvements introduced to deal with the qualification problem and the notion of plan, and (iii) the new definition and handling of threats introduced by the qualification problem.

2.3.4.1 The Agents' Planning Model and Arguments

The planning model of each agent is based on a set of literals Lit, such that $\ell \in$ Lit is a ground atom and $\sim\ell \in$ Lit is a negated ground atom, where $\sim$represents the strong negation and $\overline{\ell} = \sim\ell$. Each agent x of the MAPA architecture is initially endowed with **a planning task** $\mathbb{M}_x = ((\Psi_x, \Delta_x), A_x, F_x, G)$ where:

1. $\Psi_x \subseteq$ Lit, represents a consistent set of true facts which describe the initial state of the task.

[1] We consider that POP is the best planning approach concerned with the dynamic multiagent nature due to the ease to join several plan proposals into a single joint plan.

2. Δ_x is a set of defeasible rules $\delta = \ell_0, \ldots, \ell_k \prec \ell_0^{'}, \ldots, \ell_k^{'}$.

3. A_x is a set of actions $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$ where $\mathsf{P}(\alpha) \subseteq \mathsf{Lit}$ is a set of preconditions and $\mathsf{X}(\alpha) \subseteq \mathsf{Lit}$ is a set of effects.

4. F_x represents a consistent set of the agent-specific preferences $F_x \subseteq \{(a, d) \mid (a \in A),\ d \in [0, 100]\}$, where the action a is preferred with the estimated interest degree d.

5. $G \subseteq \mathsf{Lit}$ is the set of common goals which have to be satisfied.

The diversity of preferences is addressed by means of agreements between the agents during the planning process. We assume that agents are fully cooperative, so they have no incentives to retain relevant information. In POP, Ψ (consistent set of agents' initial states of the task) and G are encoded as dummy actions $\{\alpha_\Psi \prec \alpha_G\}$ with $\mathsf{X}(\alpha_\Psi) = \Psi$, $\mathsf{P}(\alpha_G) = G$, and $\mathsf{P}(\alpha_\Psi) = \mathsf{X}(\alpha_G) = \emptyset$.

An **argument** $\mathcal{A}$ for $\ell \in \mathsf{Lit}$, is denoted as $\mathcal{A} = (\{\ell\}, \{\Delta^{'}\})$, where $\Delta^{'}$ is a subset of defeasible rules $\Delta^{'} \subseteq \Delta$. $\mathcal{A}$ is consistent if $\mathsf{base}(\mathcal{A}) \cup \mathcal{A}$ is non-contradictory.

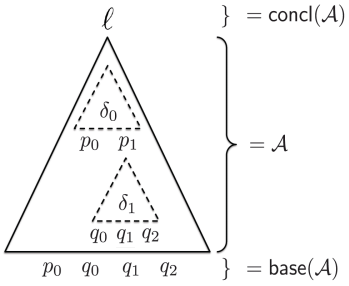

Figure 2.7: An argument $\mathcal{A}$ for l using the two defeasible rules: $\delta_0 = l \prec \{p_0, p_1\}$ and $\delta 1 = p_1 \prec \{q_0, q_1, q_2\}$.

Figure 2.39 shows an example of an argument proposed $\mathcal{A}$, where $\mathsf{literals}(\mathcal{A}) = \{l, p_0, p_1, q_0, q_1, q_2\}$. This argument for a literal ℓ does not suffice to warrant ℓ, it de-

pends on the interaction among arguments (see section 2.6.6.2), which will grant consistency. Given two arguments $\mathcal{A}, \mathcal{B}$, we say $\mathcal{A}$ *attacks* $\mathcal{B}$ if the conclusion of $\mathcal{A}$ contradicts some fact used in $\mathcal{B}$, that is, if $\overline{\mathsf{concl}(\mathcal{A})} \in \mathsf{literals}(\mathcal{B})$. Therefore, the MAPA architecture semantically differentiates between supporting arguments (or **argument steps**) as the arguments specifically used to support some open condition of the plan, and **attacking arguments** which are only introduced to attack some argument step previously introduced in the plan (i.e. it is not used to support any open condition).

2.3.4.2 The Qualification Problem and Plan Definition

The qualification problem (59), which is an important problem currently not supported in many planning architectures, is concerned with the impossibility of listing all the preconditions required for a real-world action to have its intended effect. For instance, let α (e.g. *"flying from London to Athens"*) be an action with n effects $\{e_0, e_1, \ldots\} \subseteq \mathsf{Lit}$ (e.g. e_0 =*"be at Athens city"*), which are defeated by the defeasible conditions $\{d_0, d_1, \ldots\} \subseteq \mathsf{Lit}$ (e.g. d_0 = *"Volcanic ash cloud between London to Athens"*, d_1 =*"Airport strike in London"*) respectively. Note that, if these defeasible conditions occur, the expected effects of α would not be achieved. The work in (23) solves this issue by introducing these defeasible conditions as negated preconditions of α ($\{\overline{d_0}, \overline{d_1}, \ldots\} \subseteq \mathsf{P}(\alpha)$), which must be derived by arguments.

However, an action α in MAPA architecture follows a specific representation in order to deal with this problem. We introduce a **fictitious effect** μ (meaning α *was just executed*); then we define $\mathsf{X}(\alpha) = \{\mu\}$ and expand the set of rules Δ with $\{e_k \multimap\!\prec \mu\} \cup \{\overline{e_k} \multimap\!\prec \mu, d_k\}$, where e_k represents the effect of the action α and d_k is a defeasible condition. For instance, in Figure 2.43, the precondition e_0 of the action α_G is initially derived by an argument $\mathcal{D} = (\{e_0\}, \{e_0 \multimap\!\prec \mu\})$ whose $\mathsf{base}(\mathcal{D}) = \mu$ will be satisfied by α. Then an attacking argument $\mathcal{Q} = (\{\overline{e_0}\}, \{\overline{e_0} \multimap\!\prec \mu, d_0, d_1\})$, which is a defeater of $\mathcal{D}$ ($\mathcal{Q}$ attacks $\mathcal{D}$), arises from the distributed knowledge among agents. Triangles in Figure 2.43 represent

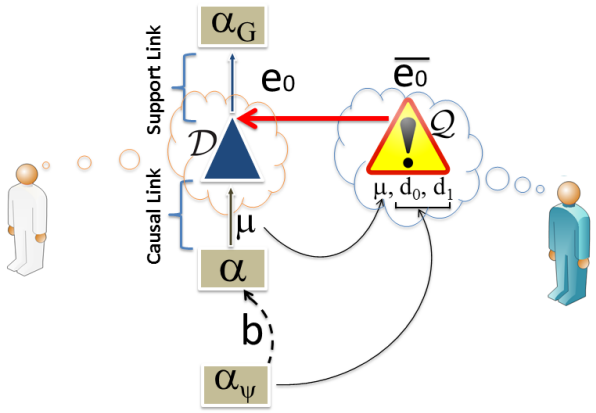

Figure 2.8: An example solving the qualification problem.

argument steps (i.e. arguments that support preconditions of action steps), for instance the argument $\mathcal{D}$, or arguments attacking some other argument, for instance the argument $\mathcal{Q}$, and both are labeled with the argument name, while rectangles represent action steps (i.e. actions that support the basis of an argument step) and are labeled with the action name.

The MAPA architecture defines **a plan** Π as a tuple $\Pi = (A(\Pi), Args(\Pi), G(\Pi), \mathcal{OC}(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi))$, where $A(\Pi)$ denotes the set of action steps, $Args(\Pi)$ represents the set of argument steps, $G(\Pi)$ is the the task's common goals, $\mathcal{OC}(\Pi)$ is a set of ordering constraints, and $\mathcal{CL}(\Pi)$ and $\mathcal{SL}(\Pi)$ represent the sets of causal and support links correspondingly. Let ℓ_1 be an open goal, motivated by some action step $\beta \in A$, i.e. $\ell_1 \in \mathsf{P}(\beta)$, and, let ℓ_2 be another open goal, motivated by some argument step $\mathcal{A} \subseteq \Delta$, i.e. $\ell_2 \in \mathsf{base}(\mathcal{A})$. Then, the goal $\ell_1 \in \mathsf{P}(\beta)$ must be supported by the argument $\mathcal{A}$, which will introduce a **support link** $(\mathcal{A}, \ell_1, \beta) \in \mathcal{SL}(\Pi)$, where $\mathcal{SL}(\Pi) \subseteq \Delta \times G(\Pi) \times A$, while the goal ℓ_2 must be satisfied by an action α, by introducing a **causal link** $(\alpha, \ell_2, \mathcal{A}) \in \mathcal{CL}(\Pi)$ where $\mathcal{CL}(\Pi) \subseteq A \times G(\Pi) \times \Delta$. Note that an argument $\mathcal{B}$ cannot support another argument $\mathcal{A}$ with a support link in $\mathcal{SL}(\Pi)$, and an action α_1 cannot support another action α_2 with a

causal link in $\mathcal{CL}(\Pi)$. To get $\mathcal{B}$ to support step $\mathcal{A}$, $\mathcal{A}$ must be replaced by $\mathcal{A} \cup \mathcal{B}$, and to get α_1 to support action step α_2, an argument $(\{e_k\}, \{e_k \prec \mu\})$ must be inserted between α_1 and α_2, where $\mathsf{X}(\alpha_2) = \mu$ and $\mathsf{P}(\alpha_1) = e_k$. Additionally, unlike in DeLP-POP, ordering constraints are placed between argument steps $(\mathcal{A}, \mathcal{B}) \in \mathcal{OC}(\Pi)$, since every action (excepting α_G) is preceded by an argument which derives its actual effects.

2.3.4.3 Interferences among Actions and Arguments

If only actions are taken into account in a planning architecture, then there is only one type of destructive interference that can arise in a plan under construction. This interference is captured by the notion of threat in POP, and occurs when a new action inserted in the plan threatens (deletes) a goal solved by other action steps. When actions and arguments are combined to construct plans, new types of interferences appear that need to be identified and resolved to obtain a valid plan. In multiagent DeLP-POP (26), we identified three types of interferences or threats, that cover all the interferences that may arise in a partial plan: argument-argument, action-argument and action-action threats.

However, since the goals must be initially derived by some argument step in the MAPA architecture, and then its basis must be satisfied by another action step (including the initial step), argument-argument threats cover all the interferences that may arise in a plan dealing with the qualification problem. Nevertheless, MAPA architecture differentiates semantically between:

1. **Planning threats** (PlaThreats): Threats that arise between two argument steps. For instance, let "w" be an open condition of the plan in Figure 2.19(c'), then the argument with an admiration is acting as a supporting argument and a PlaThreat will be discovered. These threats override the typical action-action and action-argument threats of (26). As we will discuss in subsection 2.6.6.1, this kind of threats will be discovered and possibly resolved (by promote or demote) in the *POP Search Tree*.

2. **Argumentation threats** (ArgThreats): Threats that arise when an agent discovers a new defeater which specifically attacks some argument step. Unlike the PlaThreats, here the attacks to some argument step are made by some attacking argument. For instance, in case that the argument with an admiration in Figure 2.19(c') is an attacking argument (i.e "w" is not an open goal), Figure 2.19(c') represents an ArgThreat. Although this kind of threat is also called argument-argument threat in (26), here we rename them to ArgThreats with the aim of distinguishing between PlaThreats and ArgThreats. As shown is subsection 2.6.6.2, these threats will be discovered and possibly resolved (by Defeat-the-defeater in Figure 2.19(c")) in the *POP Evaluation Tree*.

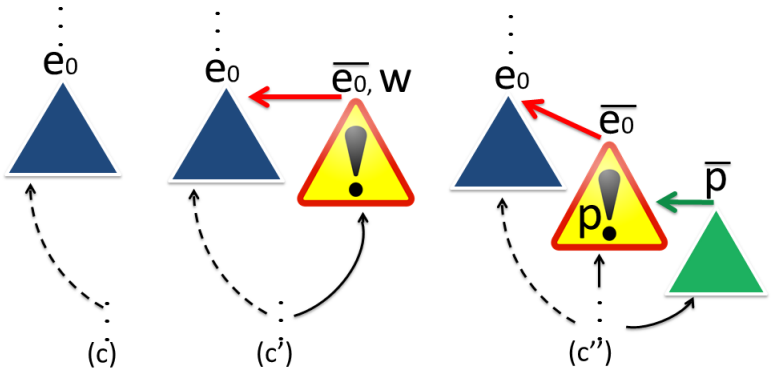

Figure 2.9: (c) Selected plan. (c') Threat. (c") Solution to (c'): Defeat-the-defeater.

2.3.5 Cooperative Distributed Planning Protocol in the MAPA Architecture

Figure 2.31 illustrates the planning protocol, which is mainly composed of three different cooperative distributed processes among the planning agents: Plan Generation, Plan Evaluation, and Plan Selection.

Different planning heuristics such as Z-LIFO (60), or the threat detect-&-solve (23) can be used to select the next open goal to solve. In our case, we will consider turn-based dialogues, a mechanism traditionally used in cooperative scenarios where agents only participate during their turn. Additionally, agents can also be modeled to put a veto on information or decisions of other agents. Agents are enumerated, and each process is implemented through a different argumentative dialogue.

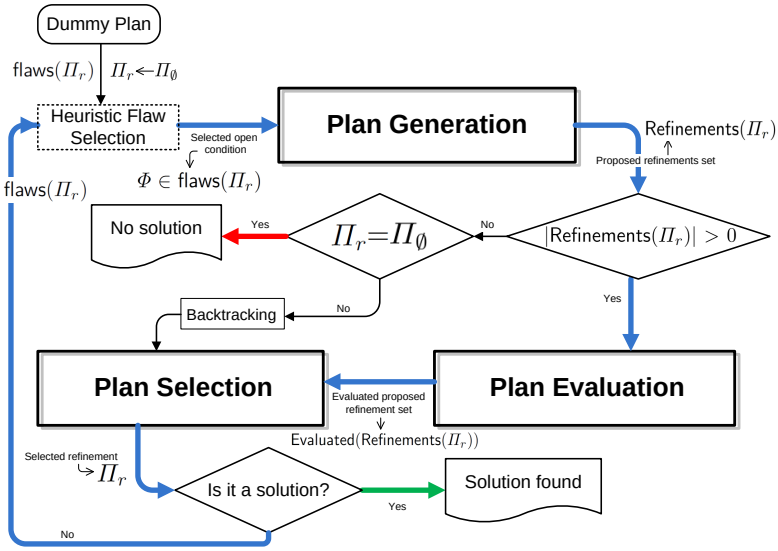

Figure 2.10: Planning Protocol in the MAPA architecture

2.3.5.1 Plan Generation

The input is both the selected plan Π_r and the selected open goal (flaw) Φ, according to the Plan Selection process (see subsection 2.6.6.3) and open goal selection heuristic. The flaw Φ can be referred to both goals and PlaThreats. The main goal of this process is to allow agents to propose **a set of refinement plans** $\mathsf{Ref}(\Pi, \Phi)$, where each $\Pi_r(\xi) \in$

$\mathsf{Ref}(\Pi, \Phi)$ is a refinement step in the *POP Search Tree* that solves a selected flaw Φ such that $\Phi \in \mathsf{flaws}(\Pi_r)$ and $\Phi \notin \mathsf{flaws}(\Pi_r(\xi))$. Following, we explain the two steps involved in this process:

1. PROPOSALS ROUND: Each agent, at its turn, proposes alternative ways to achieve or derive Φ. The process ends when all agents have had a turn. Refinements of a plan Π_r are labeled as $\Pi_r^{(n,i)}(\xi)$, where $n \in \mathbb{Z}$ indicates the refinement proposal by the agent, $i \in \mathbb{Z}$ represents the agent, and $r \in \mathbb{Z}$ represents the selected plan by the Plan Selection process. Note that, at each turn, an agent can propose as many plans as possible from its knowledge.

2. LEARNING ROUND: Each agent updates its set of actions with the new actions which appear in the refinements proposed by other agents.

The output of this process is a set of plans $\mathsf{Ref}(\Pi, \Phi)$ where each $\Pi_r(\xi) \in \mathsf{Ref}(\Pi, \Phi)$ extends Π_r. If $|\mathsf{Ref}(\Pi, \Phi)| > 0$, i.e. there is at least one refinement plan, it is used as an input to the Plan Evaluation process (see section 2.6.6.2). If $|\mathsf{Ref}(\Pi, \Phi)| = 0$, i.e. there is not any proposal to solve the flaw Φ, a backtracking step is performed, pruning the current base plan Π_r.

2.3.5.2 Plan Evaluation

Roughly, the problem stems from different agents discussing about a given plan; since these agents may have different initial facts and defeasible rules they may not agree on the evaluation of the plan at some step. Along with the *POP Search Tree* of the previous section, the MAPA architecture also considers the notion of *POP Evaluation Tree*.

Definition 1. *POP Evaluation Tree: Let $\Pi_r(\xi)$ be a refinement of plan Π_r from the previous process. A POP Evaluation Tree for $\Pi_r(\xi)$, denoted $\mathcal{T}_{\Pi_r(\xi)}$, where there is at least one argument step $(\mathcal{A}, \ell, \beta) \in \mathcal{SL}(\Pi_r(\xi))$, is defined as follows:*

- *The root of the tree is labeled with the plan $\langle \Pi_r(\xi) \rangle$.*

- *Each node of the first level* $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta))\rangle$*, is a new plan extending* $\Pi_r(\xi)$ *with some new defeater* $\mathcal{B}$ *that attacks* $\mathcal{A}$*, discovering a new* ArgThreat *in* $\Pi_r(\xi)$.

- *Each node of the second level* $\langle \Pi_r(\xi, \xi^{'}, \xi^{''}) \mid \xi^{''} = \mathsf{Defeater}(\mathcal{C}, \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta)))\rangle$*, is a new plan extending* $\Pi_r(\xi, \xi^{'})$ *with some new defeater* $\mathcal{C}$ *that attacks* $\mathcal{B}$ *(Defeat-the-Defeater), solving the* ArgThreat *in* $\Pi_r(\xi)$.

The input of this process is a set $\mathsf{Ref}(\Pi, \Phi)$ of plans proposed by the agents in the previous process. Each plan $\Pi_r(\xi) \in \mathsf{Ref}(\Pi, \Phi)$ represents the root of a new *POP Evaluation Tree* $\mathcal{T}_{\Pi_r(\xi)}$. Following, we explain the steps involved in this cooperative process:

1. ATTACK ROUND: It initiates an evaluation dialogue for the root plan of each $\mathcal{T}_{\Pi_r(\xi)}$, where each agent sends as many $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta))\rangle$ as they know at their turn (Figure 2.11). If the agent does not know how to attack a root plan, then it will skip its turn.

2. DEFENSE ROUND: It allows the agents to propose ways $\langle \Pi_r(\xi, \xi^{'}, \xi^{''}) \mid \xi^{''} = \mathsf{Defeater}(\mathcal{C}, \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta)))\rangle$ to solve discovered ArgThreats in each $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta))\rangle$. This round only applies to those *POP Evaluation Trees* which have discovered threats (Figure 2.11).

3. LEARNING ROUND: In this stage, each agent will update its sets of initial facts and defeasible rules, by extracting literals $\ell \in \mathsf{Lit}$ and defeasible rules, from arguments' bases and plan proposals. Unlike the previous Plan Generation process, where agents learn abilities as actions, this step is focused exclusively on the literals and defeasible rules.

4. EVALUATION: This stage marks each plan $\Pi_r(\xi) \in \mathsf{Ref}(\Pi, \Phi)$ as an undefeated plan, in case that defeater plans $\langle \Pi_r(\xi, \xi^{'}) \mid \xi^{'} = \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta))\rangle$ have not been discovered, or if they have been discovered but there is a plan $\langle \Pi_r(\xi, \xi^{'}, \xi^{''}) \mid$

$\xi^{''} = \mathsf{Defeater}(\mathcal{C}, \mathsf{Defeater}(\mathcal{B}, (\mathcal{A}, \ell, \beta)))\rangle$. Otherwise, $\Pi_r(\xi)$ is marked as a defeated plan.

The process ends when all the plans in $\mathsf{Ref}(\Pi, \Phi)$ have been evaluated. The output of this process is $\mathsf{Eval}(\mathsf{Ref}(\Pi_r))$, **the set of evaluated plans** (Figure 2.11). As shown in the next process, undefeated plans, which constitute the most promising refinements to reach a solution, are preferred to defeated plans. However, defeated plans are kept, since each non-resolved attack could be resolved in a subsequent evaluation process.

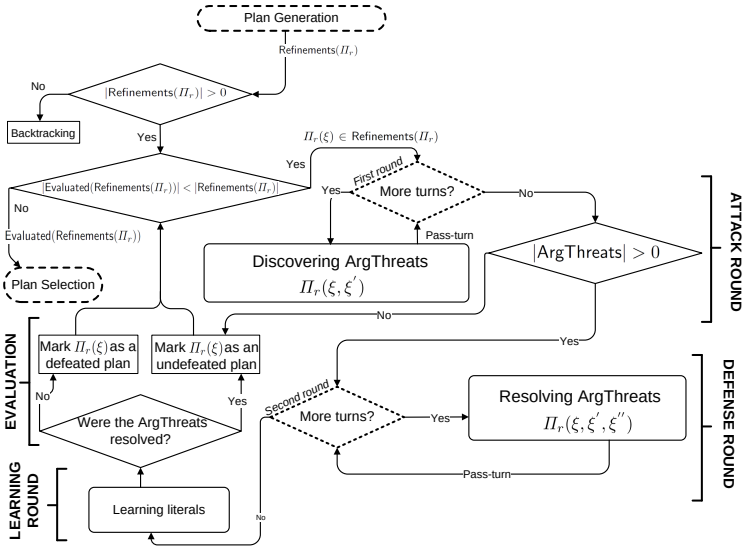

Figure 2.11: Plan Evaluation protocol overview.

2.3.5.3 Plan Selection

Plan selection can be done through the application of standard domain-independent heuristics for evaluating plans. These heuristics approximate the cost of a solution plan in terms of the number of actions, the cost or the duration of the actions. Using this type of heuristics as the *standard rating* (R_S) for plan assessment will ignore the dynamic multi-agent

nature of the MAPA architecture, where a set of preferences is assumed by each agent. Therefore, a second rating based on the agents' preferences is necessary. We will refer to it as the *preference rating* (R_F). Moreover, a third rating in terms of trust in cooperative planning is justified in (61) as a judgement about the risk attached to each component in the plan requiring cooperation, which we will call *trust rating* (R_T). R_T depends on the trust in (i) each argument step and (ii) each action step in the plan, where (i) is the trustworthiness (reputation) of the information sources which are used by the agent in order to have a perception of the environment (coded as facts and defeasible rules (23)), and, (ii) is the result of dividing the number of times the action is successfully executed into the total number of executions of the action. The MAPA architecture stores the execution of each action as a new case (62), recorded as successful if the action is executed correctly, or failure if the action failed during the execution. The success or failure of an action is determined by the achievement of the action effects. For simplicity, we only consider trust in action steps.

Unlike the Plan Generation and the Plan Evaluation process, where agents reason about agent facts, defeasible rules and actions, here agents reason about standard ratings, preference ratings, and trust ratings, considering a compromise between the desire to minimize the computational overhead and that of maximizing the quality of the plan. This process receives as input the set of evaluated plans $\mathsf{Eval}(\mathsf{Ref}(\Pi_r))$ from the Plan Evaluation process and a set of previously not-selected partial plans $\mathsf{OtherRef}$, in order to select a new plan $\Pi_r \in \{\mathsf{Eval}(\mathsf{Ref}(\Pi_r)) \bigcup \mathsf{OtherRef}\}$ as output. Following, we explain the steps involved in this process:

1. PLAN FILTERING: The aim is to guide the plan search by selecting the best subset $\mathsf{FilteredPlans} \subseteq \{\mathsf{Eval}(\mathsf{Ref}(\Pi_r)) \bigcup \mathsf{OtherRef}\}$ (as candidate plans), according to the highest $R_S(\Pi)$ and $R_T(\Pi)$, such that $\Pi \in \{\mathsf{Eval}(\mathsf{Ref}(\Pi_r)) \bigcup \mathsf{OtherRef}\}$[1], where:

[1] Undefeated plans are preferred over defeated plans.

- $R_S(\Pi) = (\mathsf{cost}(\Pi) + \mathsf{heuristic}(\Pi))$, where $\mathsf{heuristic}(\Pi)$ is a heuristic estimation of the cost of reaching a solution plan $\Pi^\star$ from Π, and,
- $R_T(\Pi)$ is the product of the trust values of the action steps in Π.

2. PLAN RANKING: The agents ($i \in \{1, 2, \ldots, k\}$) calculate their preference ratio for each candidate plan $\Pi_n \in \mathsf{FilteredPlans}$. For this purpose, they review whether each action $a \in A(\Pi_n)$ is preferred by them. If an action $a_1 \in A(\Pi_n)$ is preferred ($(a_1, d_1) \in F_i \mid d_1 > 50$), then they increase by one the value $R^i_F(\Pi_n)$; if they do not prefer an action $a_2 \in A(\Pi_n)$, ($(a_2, d_2) \in F_i \mid d_2 <= 50$), then they subtract one unit from $R^i_F(\Pi_n)$, and otherwise they keep $R^i_F(\Pi_n)$ unchanged. This stage, which simulates a internal reasoning process for or against to select each plan Π_n, allows each agent to establish a preference relation between the plans in $\mathsf{FilteredPlans}$.

3. PLAN NEGOTIATION: Since each agent has identified its preferred candidate plans, now the purpose of the negotiation is to reach an **agreement** about the next base plan $\Pi_r \in \mathsf{FilteredPlans}$. This stage can range from a simple voting process to a more sophisticated negotiation mechanism.

Finally, $\{\mathsf{NonSelectedPlans} \subset \mathsf{Eval}(\mathsf{Ref}(\Pi_r)) \mid \Pi_r \notin \mathsf{NonSelectedPlans}\}$ is added to the set $\mathsf{OtherRef}$, and the process returns the agreed plan Π_r. If Π_r is not a solution, the control will be passed to the Heuristic Flaw Selection (see Figure 2.31). Otherwise, the planning process will end successfully.

2.3.6 Evaluating the MAPA architecture within the context of a transit journey planning service

Transit users generally know their origin and destination cities. Based on the schedules provided by the transit agencies, users choose the best routes that match their travel needs.

For this purpose, a Transit Journey Planning Service (TJPS) (a specialized electronic search engine) is used to find the best route between two locations by using some means of transportation. TJPSs are being widely used by transit agencies accessed through a web user interface on a computer terminal to support clients' requests on public transport information. Most of the existing TJPSs, provided by transit agencies and companies (Google Transit Planner, Transport Direct, Transport for London, Trip Planning Tool etc.)[1], are based on static schedule data. To the best of our knowledge, these centralized planners (i) do not react to environmental changes such as bad weather, traffic jams or bad railroads, and therefore they do not provide support to defeasible reasoning, and (ii) are not able to work in a cooperative distributed environment so there is no choice for exchanging information between them.

However, defeasible reasoning is becoming an increasingly important feature in many environments where context awareness in not fully specified. The work in (63) presents a potential application for distributed defeasible reasoning in ambient computing environments, where the ambient agents, who have different viewpoints, have to face the available context. Similarly, defeasible reasoning is also being applied to semantic web and e-commerce (64). Here, we present a novel application of cooperative distributed defeasible planning to a TJPS problem.

The MAPA architecture is implemented in Magentix 2[2], a platform for open Multi-agent Systems based on the Apache Qpid[3] implementation of AMQP [4] for communication between agents. This platform incorporates a security module which provides key features regarding security, privacy, openness and interoperability not offered by other current agent platforms.

[1] `http://www.google.com/transit`, `http://www.transportdirect.info`, `http://www.journeyplanner.org`, `http://www.networkedtraveler.org`

[2] `http://www.gti-ia.dsic.upv.es/sma/tools/magentix2/index.php`

[3] `http://qpid.apache.org/`

[4] `http://www.amqp.org`

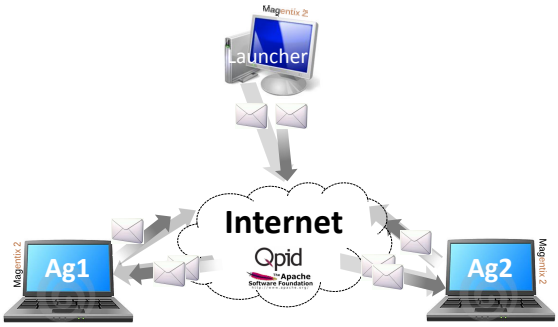

Figure 2.12: Deploying the MAPA architecture.

2.3.6.1 Preliminaries

According to the multi-agent systems paradigm, we have implemented a CIS as a collection of software agents (Figure 2.12) in the MAPA architecture. Each agent simulates an information system, and interacts with the others so as to achieve the common goals, thus forming a multi-agent society. More specifically, we consider a scenario with six different cities and two geographically distributed transit agencies, $Ag1$ (Greece transit agency) and $Ag2$ (UK transit agency), aimed at providing a customer with a plan to travel from $London$ to $Athens$ (Figure 2.13). The agencies are implemented as agents and have different knowledge (knowledge is fully distributed), so two pieces of information derived from each agent may appear to be contradictory. There are several ways to travel between both cities: via car, ship, train or plane. Let's assume that $Ag1$ uses *BBC News* as a source of information, but $Ag2$ prefers *CNN News* to keep up to date, and both agree on finding a plan that minimizes the journey duration. The planning tasks of the agents are defined in Figure 2.22, where we consider propositional STRIPS (45) planning representation.

In what follows, we define the meaning of each literal and action. Literals:

- A, L - Athens, London; B, C, D, F - Other cities,

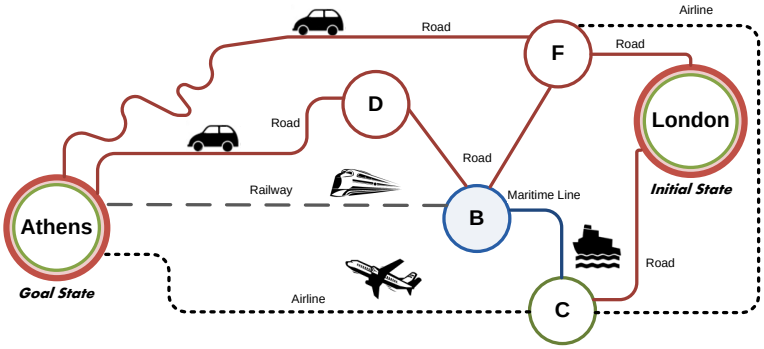

Figure 2.13: Scenario of the application example.

- Ag, car, tra, pl, shi - a customer, a car, a train, a plane, a ship,
- r, rl, al, ml - a road, a railway, an airline company, a maritime line,
- bw, sn, wg, va, ds, aeo - bad weather, snow, wind gusts, volcano ash cloud, dangerous situation, airplane engines work well (after test),
- br, ll, esf, fp - bad railroad, landslides, electrical supply failure, flying panic,
- h, tj, $kudBBC$, $kudCNN$ - holidays, traffic jam, kept up to date by BBC news, kept up to date by CNN news, and,
- μ_C, μ_P, μ_T, μ_S - moved car, moved plane, moved train and moved ship.

Actions are the following (notation: $\mathsf{X}(\alpha) \overset{\alpha}{\leftarrow} \mathsf{P}(\alpha)$, i.e. the action effects are indicated on the left side, while the action preconditions on the right side):

1. $mP(pl, x, y)$: moving plane 'pl' from location 'x' to 'y' takes 2 time units and 400 cost units.
2. $mT(tra, x, y)$: moving train 'tra' from location 'x' to 'y' takes 6 time units and 200 cost units.

$$\Psi_{Ag1} = \left\{ \; (wg\ B\ A);\ aeo;\ kudBBC; (at\ \mathsf{Ag}\ L); fp;\ (at\ pl\ C);\ (at\ car\ L);\ (link\ al\ C\ A);\ (link\ r\ D\ A); \ldots \; \right\}$$

$$\Psi_{Ag2} = \left\{ \; kudCNN;\ fp;\ (at\ \mathsf{Ag}\ L);\ (at\ tra\ B);\ (at\ shi\ C)\ (link\ rl\ B\ A);\ (link\ ml\ C\ B);\ (link\ r\ F\ A); \ldots \; \right\}$$

$$\Delta_{Ag1} = \left\{ \begin{array}{c} \{(at\ pl\ ?y), (at\ \mathsf{Ag}\ ?y)\} \prec (\mu_P\ ?x\ ?y);\ \{\sim(at\ tra\ ?y), \sim(at\ \mathsf{Ag}\ ?y)\} \prec \{(\mu_T\ ?x\ ?y), (br\ ?x\ ?y)\}; \\ \{(at\ car\ ?y), (at\ \mathsf{Ag}\ ?y)\} \prec (\mu_C\ ?x\ ?y);\ \{\sim(at\ shi\ ?y), \sim(at\ \mathsf{Ag}\ ?y)\} \prec \{(\mu_S\ ?x\ ?y), (ss\ ?x\ ?y)\}; \\ (br\ ?x\ ?y) \prec (esf\ ?x\ ?y);\ (esf\ ?x\ ?y) \prec (sn\ ?x\ ?y);\ (sn\ B\ A) \prec kudBBC;\ \sim(va\ C\ A) \prec aeo; \ldots \end{array} \right\}$$

$$\Delta_{Ag2} = \left\{ \begin{array}{c} \{\sim(at\ pl\ ?y), \sim(at\ \mathsf{Ag}\ ?y)\} \prec \{(\mu_P\ ?x\ ?y), (ds\ ?x\ ?y)\};\ \{(at\ tra\ ?y), (at\ \mathsf{Ag}\ ?y)\} \prec (\mu_T\ ?x\ ?y); \\ \{\sim(at\ car\ ?y), \sim(at\ \mathsf{Ag}\ ?y)\} \prec \{(\mu_C\ ?x\ ?y), (tj\ ?x\ ?y)\};\ \{(at\ shi\ ?y), (at\ \mathsf{Ag}\ ?y)\} \prec (\mu_S\ ?x\ ?y); \\ (ds\ ?x\ ?y) \prec (va\ ?x\ ?y);\ (va\ C\ A) \prec kudCNN;\ \sim(ll\ ?x\ ?y) \prec \sim(bw\ ?x\ ?y); \\ \sim(ll\ ?x\ ?y) \prec \sim(bw\ ?x\ ?y);\ \sim(bw\ B\ A) \prec kudCNN;\ \sim(sn\ B\ A) \prec kudCNN; \\ (tj\ ?x\ ?y) \prec \{(h\ ?x)(link\ r\ ?x\ ?y)\};\ (h\ F) \prec kudCNN;\ \ldots \end{array} \right\}$$

$$A_{Ag1} = \left\{ \begin{array}{l} 1.\ (\mu_C\ ?x\ ?y) \xleftarrow{fMc} \{(link\ r\ ?x\ ?y),\ (at\ car\ ?x),\ (at\ \mathsf{Ag}\ ?x)\} \\ 2.\ (\mu_P\ ?x\ ?y) \xleftarrow{mP} \{(link\ al\ ?x\ ?y),\ (at\ pl\ ?x),\ (at\ \mathsf{Ag}\ ?x)\} \end{array} \right\}$$

$$A_{Ag2} = \left\{ \begin{array}{l} 3.\ (\mu_T\ ?x\ ?y) \xleftarrow{mT} \{(link\ rl\ ?x\ ?y),\ (at\ tra\ ?x),\ (at\ \mathsf{Ag}\ ?x)\} \\ 4.\ (\mu_S\ ?x\ ?y) \xleftarrow{mS} \{(link\ ml\ ?x\ ?y),\ (at\ shi\ ?x),\ (at\ \mathsf{Ag}\ ?x)\} \end{array} \right\}$$

$$F_{Ag2} = \left\{ \; (mT,\ 90);\ (mP\ 0);\ (mS\ 60) \; \right\}$$

$$F_{Ag1} = \left\{ \; (fmC\ 70);\ (mT,\ 80);\ (mP\ 5) \; \right\}$$

$$G = \{(at\ \mathsf{Ag}\ A)\}$$

Figure 2.14: Initial facts, defeasible rules, actions, preferences and common goals.

3. $mS(shi, x, y)$: moving ship 'shi' from location 'x' to 'y' takes 3 time units and 100 cost units.

4. $fMc(car, x, y)$: fast-moving car 'car' from location 'x' to 'y' takes 8 time units and 80 cost units.

2.3.6.2 Implementation

The planning process starts with an empty plan $\Pi_\emptyset = \{\alpha_\Psi \prec \alpha_G\}$ and $\mathsf{flaws}(\Pi_\emptyset) = \{(at\ \mathsf{Ag}\ A)\}$. First, the MAPA architecture enters the **Plan Generation** process, where four plans are suggested: i) taking the car between D and A, $\Pi_\emptyset^{(1,Ag1)}(\xi)$, or ii) between F and A, $\Pi_\emptyset^{(2,Ag1)}(\xi)$, iii) taking the train between B and A, $\Pi_\emptyset^{(1,Ag2)}(\xi)$, and iv) taking the plane between C and A, $\Pi_\emptyset^{(3,Ag1)}$. We only show the plan iv) $\Pi_\emptyset^{(3,Ag1)}(\xi) = \{(mP,\ (\mu_P\ C\ A), \mathcal{A}^{\text{Ag1}}), (\mathcal{A}^{\text{Ag1}}, (at\ \mathsf{Ag}\ A), \alpha_G)\}$ where $\mathcal{A}^{\text{Ag1}} = (\{(at\ \mathsf{Ag}\ A)\}, \{(at\ \mathsf{Ag}\ A) \prec (\mu_P\ C\ A)\}$ (see Figure 2.16(a)). The agents learn the actions they did not know from these plans.

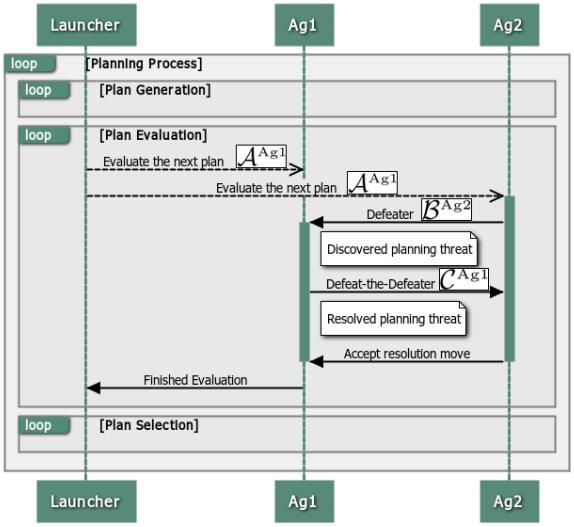

Figure 2.15: Discussing about the plan $\Pi_\emptyset^{(3,Ag1)}(\xi)$ in the Plan Evaluation process.

Second, the **Plan Evaluation** process starts, where: i) $\Pi_\emptyset^{(1,Ag1)}(\xi)$ is not attacked by any defeater and it is labeled as an undefeated plan. ii) $\Pi_\emptyset^{(2,Ag1)}(\xi)$ is attacked because city F is on holiday, so a traffic jam can be expected in the road between F and A, and, therefore, the effects of the action fMc may not be satisfied. Since there are not proposals to solve this ArgThreat, $\Pi_\emptyset^{(2,Ag1)}(\xi)$ is labeled as a defeated plan. iii) $\Pi_\emptyset^{(1,Ag2)}(\xi)$ receives one attack because snow is expected between B and A, so an electrical failure that damages the railroad between B and A might occur. If this happens, the effects of the action mT may not be satisfied. Here, $Ag2$ proposes a Defeat-the-defeater, which

justifies that snow conditions are not expected between B and A, and then $\Pi_{\emptyset}^{(1,Ag2)}(\xi)$ is labeled as undefeated plan. iv) $Ag2$ attacks $\Pi_{\emptyset}^{(3,Ag1)}(\xi)$ with $\langle \Pi_{\emptyset}^{(3,Ag1)}(\xi,\xi') \mid \xi' = \mathsf{Defeater}(\mathcal{B}^{\mathrm{Ag2}}, (\mathcal{A}^{\mathrm{Ag1}}, (at\ \mathsf{Ag}\ A), \alpha_G))\rangle$ where $\mathcal{B}^{\mathrm{Ag2}} = (\{\sim(at\ \mathsf{Ag}\ A)\}, \{\sim(at\ \mathsf{Ag}\ A) -\prec\{(\mu_P\ C\ A), (ds\ C\ A)\}; (ds\ C\ A) \multimap (va\ C\ A);\ (va\ C\ A) \multimap kudCNN\})$ (see Figure 2.15) because the volcano ashes are expected between the city C and A according to the *CNN News*, but $Ag1$ moves against $(ds\ C\ A)$ with $\langle \Pi_{\emptyset}^{(3,Ag1)}(\xi,\xi',\xi'') \mid \xi'' = \mathsf{Defeater}(\mathcal{C}^{\mathrm{Ag1}}, \mathsf{Defeater}(\mathcal{B}^{\mathrm{Ag2}}, (\mathcal{A}^{\mathrm{Ag1}}, (at\ \mathsf{Ag}\ A), \alpha_G)))\rangle$ where $\mathcal{C}^{\mathrm{Ag1}} = (\{\sim(va\ C\ A)\}, \{\sim(va\ C\ A) \multimap aeo\})$ (see Figure 2.15). It is a Defeat-the-defeater resolution move since $\sim \mathsf{concl}(\mathcal{C}^{\mathrm{Ag1}}) \in \mathsf{literals}(\mathcal{B}^{\mathrm{Ag2}})$ (see Figure 2.16(a")), and then $\Pi_{\emptyset}^{(3,Ag1)}(\xi)$ is labeled as an undefeated plan. The agents learn the literals and defeasible rules, they do not know at the beginning of the turn.

Third, the **Plan Selection** process starts. The best subset of plans is defined as $\mathsf{FilteredPlans} = \{\Pi_{\emptyset}^{(1,Ag2)}(\xi), \Pi_{\emptyset}^{(3,Ag1)}(\xi)\}$, since $\Pi_{\emptyset}^{(2,Ag1)}(\xi)$ was labeled as a defeated plan and $\mathsf{heuristic}(\Pi_{\emptyset}^{(1,Ag1}(\xi))$ returns a high value. Finally, agents choose $\Pi_r = \Pi_{\emptyset}^{(1,Ag2)}(\xi)$ as they prefer to take the train because of their fear of flying. For space reasons, we omit the rest of the plan search.

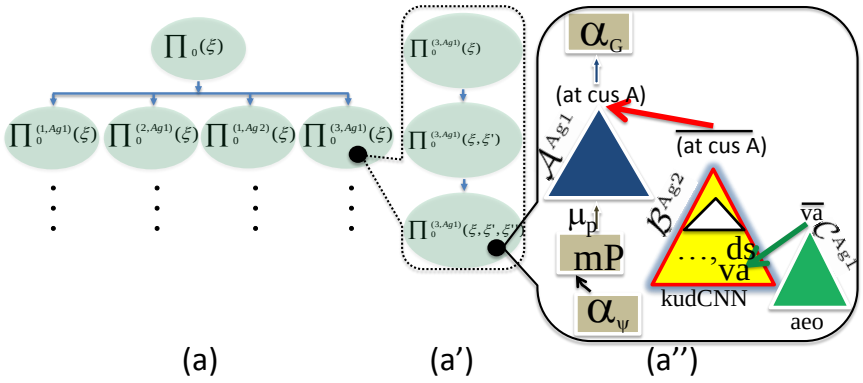

Figure 2.16: Screenshots: (a) The *POP Search Tree*. (a') The *POP Evaluation Tree* for the plan $\Pi_{\emptyset}^{(3,Ag1)}(\xi)$. (a") Viewing the content of the plan $\Pi_{\emptyset}^{(3,Ag1)}(\xi,\xi',\xi'')$.

2.3.7 Conclusions and Future work

We have presented MAPA, a decentralized architecture for cooperative planning in multiagent DeLP-POP, dealing with the qualification problem. It is implemented as three independent cooperation processes between agents of a team who propose, criticize, defend and select alternative plans by means of arguments and actions. For future work, we intend to work in several directions: extending MAPA to other multi-agent scenarios like argumentation-based negotiation or to temporal planning (65); and evaluating MAPA in applications of dynamic networked cooperative business processes and knowledge-sharing, including the ability to work with and within complex supply chains. Finally, evaluating the efficiency and effectiveness of the MAPA architecture.

Acknowledgments.

This work is mainly supported by FPU grant reference AP2009-1896 awarded to Sergio Pajares Ferrando and projects TIN2008-06701-C03-01, Consolider Ingenio 2010 CSD2007-00022, and PROMETEO/2008/051.

2.4 Selected paper 3: Multiagent Argumentation for Cooperative Planning in DeLP-POP (AAMAS 2011) and its extension in (ArgMAS 2012)

Abstract. *This contribution proposes a model for argumentation-based multi-agent planning, with a focus on cooperative scenarios. It consists in a multi-agent extension of* DeLP-POP*, partial order planning on top of argumentation-based defeasible logic programming. In* DeLP-POP*, actions and arguments (combinations of rules and facts) may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. In a cooperative planning problem a team of agents share a set of goals but have diverse abilities and beliefs. In order to plan for these goals, agents start a stepwise dialogue consisting of exchanges of plan proposals, plus arguments against them. Since these dialogues instantiate an A^* search algorithm, these agents will find a solution if some solution exists, and moreover, it will be provably optimal (according to their knowledge).*

2.4.1 Introduction

Artificial Intelligence Planning is the task of building control algorithms that synthesize a course of action achieving a desired set of goals, given some (information about the) initial state. Partial-order planning (POP) (43) is a planning paradigm based on the principle of *least commitment*: decisions on parameter bindings and action orderings are not committed until necessary during the construction of the plan. The POP approach is one of the more open planning frameworks as evidenced by the fact that most existing architectures for integrating planning with execution, information gathering, and scheduling are based on partial order planners (54, 55, 56). Even for simple planning tasks, partial order planners offer a higher degree of execution flexibility.

Unlike classical planning, in many real-world applications agents can disagree about which facts can be inferred to hold in some state or after executing some plan. For this purpose, argumentation, which has recently become a very active research field in computer science (5, 37, 39), can be viewed as a powerful tool to reason about inconsistent beliefs through a comparison between arguments for and against some conclusion. More specifically, Defeasible Logic Programming (DeLP) (24) is a framework for reasoning

about defeasible information (also known as defeasible reasoning), where tentative conclusions, obtained from incomplete information, can later be deemed no longer valid after new information becomes available. Nowadays, DeLP is a well known framework for dealing with defeasible reasoning.

Argumentative methods, indeed, can also apply to the process of planning search, making this process deliberative. In this direction, an extension of POP with DeLP-style argumentation, denoted DeLP-POP framework, was introduced in (23) for single-agent planning, where both actions and arguments may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. Unlike actions, arguments will not only be introduced to intentionally support some step of a plan, but they will also be presented to defeat or defend other supporting arguments in the plan. The advantages of DeLP-POP towards reasoning about actions are clear: if planning techniques prevent the well-known frame problem, by getting rid of the need to explicitly represent what does not change after an action, DeLP-POP succeeds against the qualification problem as well, since DeLP-rules can be used to encode defeasible effects of actions, as shown in Section 2.4.2.3. When actions and arguments are combined in a partial order plan, new types of interferences or threats appear (23). These interferences need to be identified and resolved to obtain valid plans.

Although planning has been, and is, extensively studied in single-agent environments, nowadays, more and more real-world applications require a planning environment with more than one agent. This is where multi-agent planning methods come into play. These enable the agents to reason about their interactions and ensure that their individual partial plan proposals are efficient and effective. Thus, multi-agent planning generalizes the problem of planning in domains where several agents plan and act together and have to share resources, activities, and goals.

The present contribution proposes a formal model of argumentative dialogues for multi-agent planning, with a focus on cooperative planning. It consists in a multi-agent extension of the DeLP-POP framework. Here, we have a team of agents aware of a common set of goals (hence trustable), but ignorant of others' abilities and beliefs, who must find a plan. An obvious solution, centralized planning carried by some planner with knowledge of these agents' beliefs and actions, would arise questions of efficiency and privacy loss (beyond necessity).

The main challenge presented by cooperative multi-agent DeLP-POP is plan evalua-

tion and search. We will use centralized DeLP-POP just for comparison with dialogues proposed. A dialogue consists in a series of exchanges of (1) plan proposals addressing the current goal, plus (2) potential arguments against (1). Atomic information (facts, rules, actions) contained in others' messages (1) and (2) will be extracted and adopted to devise new ideas for both (1) and (2). Dialogues are turn-based, since this choice models typically cooperative scenarios where all agents are treated in a uniform way, but also can (by adding some restrictions) model agents with power to veto information or decisions.

The main result of this contribution is that such a dialoguing team of planner agents actually implements an A^* search procedure. Thus, the team of agents need not search the full space of plans: the dialogue terminates at a solution (if some solution exists) which is provably optimal.

This paper is a revised version of the work titled *Multiagent Argumentation for Cooperative Planning in* DeLP-POP (26). It is organized as follows: section 2 gives some preliminaries and notations; section 3 formalizes some concepts; section 4 presents the argumentative dialogues to Multi-Agent Planning; section 5 shows an example of application to validate the framework. Finally, we conclude and present some directions for future work.

2.4.2 Preliminaries

Notation: Throughout the paper we make use of these conventions: the projection functions are $\pi_k(\langle a_0, \ldots, a_n\rangle) == a_k$ (for $k \leq n$), and $\pi_{\widehat{k}}(\langle a_0, \ldots, a_{k-1}, a_k, a_{k+1}, \ldots, a_n\rangle) = \langle a_0, \ldots, a_{k-1}, a_{k+1}, \ldots a_n\rangle$. Given propositional variables $p, \ldots \in \mathsf{Var}$, and a negation $\sim$, we define the set of literals $\ell \in \mathsf{Lit} = \mathsf{Var} \cup \{\sim p \mid p \in \mathsf{Var}\}$. Also, define $\overline{\ell}$ as $\overline{p} = \sim p$, and $\overline{\sim p} = p$, for any $p \in \mathsf{Var}$; and for $X \subseteq \mathsf{Lit}$, $\overline{X} = \{\overline{\ell} \mid \ell \in X\}$. In general, if $F : X \to Y$ is a function and $X' \subseteq X$, we denote $F[X'] = \{f(x) \mid x \in X'\}$. The transitive closure of a relation R is denoted $\mathsf{tc}(R)$. The size of a set X is denoted $|X|$. If X is a set, $\mathcal{P}(X)$ denotes its power set, and $X\binom{\sigma\tau\cdots}{\sigma'\tau'\ldots}$ denotes the set obtained by replacing σ by σ', τ by τ', ... in set X.

2.4.2.1 DeLP: Defeasible Logic Programming

In (24), the authors propose a non-monotonic consequence relation, called *warrant*, built upon the relation of defeat between constructible arguments for or against a literal. A

defeasible logic program (or *de.l.p.*, henceforth) is a pair $T = (\Psi, \Delta)$ consisting of a strict and a defeasible part:

- a consistent set $\Psi \subseteq \mathsf{Lit}$ of *facts*, and
- a set Δ of defeasible *rules* $\delta = \ell \prec\!\!- \ell_0, \ldots, \ell_k$

where $\ell, \ell_0, \ldots, \ell_k \subseteq \mathsf{Lit}$. Rule $\ell \prec\!\!- \ell_0, \ldots, \ell_k$ expresses: warrant for $\ell_0, \ldots, \ell_n$ provide a (defeasible) reason for ℓ to be warranted[1]. We denote $\mathsf{body}(\delta) = \{\ell_0, \ldots, \ell_n\}$ and $\mathsf{head}(\delta) = \ell$ as, respectively, the *body* and *head* of δ.

Derivability in $T = (\Psi, \Delta)$ is closure under *modus ponens*: literals in Ψ are derivable and, given a rule δ, if each $\ell \in \mathsf{body}(\delta)$ is derivable, then $\mathsf{head}(\delta)$ is derivable.
An *argument* $\mathcal{A}$ for ℓ in a de.l.p. (Ψ, Δ), denoted $\langle \mathcal{A}, \ell \rangle$ or simply $\mathcal{A}$, is a set of rules $\mathcal{A} \subseteq \Delta$ such that (i) ℓ is derivable from $(\Psi, \mathcal{A})$, (ii) the set $\Psi \cup \mathcal{A}$ is non-contradictory, and (iii) $\mathcal{A}$ is a minimal subset of Δ satisfying (i) and (ii).

We also define, for an argument $\mathcal{A}$ for ℓ

$$\begin{aligned} \mathsf{concl}(\mathcal{A}) &= \ell, \\ \mathsf{base}(\mathcal{A}) &= (\bigcup \mathsf{body}[\mathcal{A}]) \setminus \mathsf{head}[\mathcal{A}], \text{ and} \\ \mathsf{literals}(\mathcal{A}) &= (\bigcup \mathsf{body}[\mathcal{A}]) \cup \mathsf{head}[\mathcal{A}] \end{aligned}$$

A derivation of -or argument for- a literal ℓ from (Ψ, Δ), still, does not suffice for its being warranted in (Ψ, Δ). The latter depends on the interaction among arguments, which will grant consistency.

Given two arguments $\mathcal{A}, \mathcal{B}$, we say $\mathcal{A}$ *attacks* $\mathcal{B}$ if the conclusion of $\mathcal{A}$ contradicts some fact used in $\mathcal{B}$, that is, if $\overline{\mathsf{concl}(\mathcal{A})} \in \mathsf{literals}(\mathcal{B})$. This attack relation may roughly be seen as symmetric, in the sense that each attacked argument $\mathcal{B}$ contains a sub-argument $\mathcal{B}'$ attacking $\mathcal{A}$. (A *sub-argument* of $\mathcal{B}$ is a subset $\mathcal{B}' \subseteq \mathcal{B}$ supporting some inner conclusion ℓ' of $\mathcal{B}$, i.e. with $\ell' \in \mathsf{literals}(\mathcal{B})$.) To decide which contending argument prevails, a notion for preference among pairs of conflicting arguments is needed. The formal criterion for preference here adopted lies in a comparison of information used in each argument: an attacking argument which makes use of more precise rules (or more premises) is a *proper defeater* for -is preferred to- the contending argument. If two contending arguments are not comparable in these terms, they are a *blocking defeater* for each other. Or, less abstractly, one could instead specify some particular preference between rules and then induce a defeat relation for arguments out of it. See (66) for details.

[1] Strict rules, introduced in (66), (24), have not been considered in planning, see (23).

Given an argument $\mathcal{A}_0$ for ℓ, an *argumentation line* $\Lambda = [\mathcal{A}_0, \ldots, \mathcal{A}_n]$ in (Ψ, Δ) is a sequence of arguments constructible in (Ψ, Δ), where each argument $\mathcal{A}_{k+1}$ is a defeater for its predecessor $\mathcal{A}_k$. Some further conditions are needed to rule out circular or inconsistent argumentation lines; briefly, arguments supporting (resp. interfering with) $\mathcal{A}_0$, i.e. of the form $\mathcal{A}_{2n}$ (resp. $\mathcal{A}_{2n+1}$) must form a consistent set, and no sub-argument $\mathcal{A}'$ of an argument $\mathcal{A}_m \in \Lambda$ may appear later in Λ (i.e. it cannot be that $\mathcal{A}' = \mathcal{A}_{m'}$ with $m' > m$); see (24) and (23).

Since in a de.l.p. (Ψ, Δ) an argument can have several defeaters, different argumentation lines rooted in $\mathcal{A}_0$ can exist. Their union gives rise to a tree-like structure, the *dialectical tree* for $\mathcal{A}_0$, denoted $\mathcal{T}_{\mathcal{A}_0}(\Psi, \Delta)$. To check whether $\mathcal{A}_0$ is *defeated* or *undefeated*, the following procedure on $\mathcal{T}_{\mathcal{A}_0}(\Psi, \Delta)$ is applied: label with a *U* (for *undefeated*) each terminal node in the tree (i.e. each argument with no defeaters at all). Then, in a bottom-up fashion, we label a node with:

$$\begin{cases} U & \text{if each of its successors is labeled with a } D \\ D & \text{(for } \textit{defeated}\text{) otherwise} \end{cases}$$

Finally, we say a literal ℓ is *warranted* in (Ψ, Δ), denoted $\ell \in \mathsf{warr}(\Psi, \Delta)$, iff there exists an argument $\mathcal{A}$ in (Ψ, Δ) with $\mathsf{concl}(\mathcal{A}) = \ell$ and $\mathcal{A}$ labeled *U* in $\mathcal{T}_{\mathcal{A}}(\Psi, \Delta)$. Henceforth, $\mathcal{B}$ *defeats* $\mathcal{A}$ will stand for: $\Lambda = [\ldots, \mathcal{A}, \mathcal{B}, \ldots]$ is acceptable.

2.4.2.2 DeLP-POP: A DeLP extension for POP planning

We briefly recall here state-based and POP planning methods, before introducing DeLP-POP. A planning domain is a tuple $\mathbb{M} = (\Psi, \mathsf{A}, G)$ where $\Psi \subseteq \mathsf{Lit}$ represents initial atomic facts, A is a set of actions and $G \subseteq \mathsf{Lit}$ is the set of goals of an agent. Here, an action $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$ is a set of preconditions (for α to be applicable) and effects. A solution is a plan Π leading a Ψ-world into a G-world by means of actions $\mathsf{A}(\Pi) \subseteq \mathsf{A}$.

In state-based planning, a plan Π is a linear sequence of actions, and thus before each action α_k in $\mathsf{A}(\Pi)$, we know which consistent state $\sigma_k \subseteq \mathsf{Lit}$ will hold, with σ_k consistent.

In contrast, a partial order plan (henceforth: plan) Π is a set of actions whose execution ordering $\prec_\Pi$ (i.e. links on action pairs) is only *partially specified* (thus encoding multiple linear plans). In POP, Ψ and G are encoded as dummy actions $\alpha_\Psi \prec_\Pi \alpha_G$ with $\mathsf{X}(\alpha_\Psi) = \Psi$, $\mathsf{P}(\alpha_G) = G$ and $\mathsf{P}(\alpha_\Psi) = \mathsf{X}(\alpha_G) = \emptyset$. Partial orderings give rise to the notion of *threat* in Π: an action step *potentially* interfering with (applicability of) some other action

step. The set of all threats to a plan Π will be denoted $\mathsf{AllThreats}(\Pi)$. When detected, threats are to be solved by some threat resolution step. Thus in POP, the set of *flaws* to be solved in a plan Π includes threats and pending goals(initially being $\mathsf{AllThreats}(\Pi) = \emptyset$ and $G(\Pi) = \mathsf{P}(\alpha_G)$). The partial order of Π determines, for each $\alpha \in \mathsf{A}(\Pi)$, a (possibly inconsistent) set of facts *potentially* planned to occur before α (i.e. the threats to this α). This set, called here the proto-state of α (in Π), will be denoted S^{Π}_{α}.

An extension of POP with DeLP-style argumentation, denoted DeLP-POP, was introduced in (23). A DeLP-POP planner can appeal both to arguments and actions as a way to resolve goals or threats. The original DeLP or POP notions of argument, planning domain, plan, link and threat must be modified accordingly. An argument $\mathcal{A} \subseteq \Delta$ is consistent if $\mathsf{base}(\mathcal{A}) \cup \mathcal{A}$ *is non-contradictory* (instead of condition (ii) above for $\Psi \cup \mathcal{A}$, since now arguments may apply everywhere, not just at Ψ). DeLP-POP planning domains $\mathbb{M} = (T, \mathsf{A}, G)$ contain now a de.l.p. $T = (\Psi, \Delta)$, where the set of initial facts $\Psi \subseteq \mathsf{Lit}$ induces α_Ψ as before and the new element Δ contains defeasible rules that may apply anywhere in the plan. An *action* is a 3-tuple of the form $\alpha = \langle \mathsf{P}(\alpha), \mathsf{C}(\alpha), \mathsf{X}(\alpha) \rangle$, described by, resp., sets of preconditions, constraints and effects. If literals in $\mathsf{P}(\alpha)$ are enforced (or warranted) and those in $\mathsf{C}(\alpha)$ fail to be enforced (or warranted), then action α is *applicable* and its execution will enforce each $\ell \in \mathsf{X}(\alpha)$ (thus deleting $\overline{\ell}$ if holding previously). An argument $\mathcal{A}$ is *applicable* at S^{Π}_{α} if $\mathsf{base}(\mathcal{A})$ is enforced in S^{Π}_{α}; in this case $\mathsf{concl}(\mathcal{A})$ is derivable. See (23)'s backward planning algorithm for a full description of an *instance* κ of an action- or argument-steps, or an open goal *in a plan* Π. Each such instance κ is labeled by its full path of links up to some $g \in G$, i.e. $\langle \kappa, \ldots, g \rangle$.

Let ℓ be an open goal, motivated by some step $\beta \in \mathsf{A}(\Pi)$ or $\mathcal{A} \subseteq \Delta$; i.e. $\ell \in \mathsf{P}(\beta)$ or $\ell \in \mathsf{base}(\mathcal{A})$. If goal ℓ is planned to be enforced by an action α, this is encoded as a *causal link* of Π, in a set denoted by $\mathcal{CL}(\Pi)$: $(\alpha, \ell, \kappa) \in \mathcal{CL}(\Pi) \subseteq \mathsf{A}(\Pi) \times G(\Pi) \times (\mathsf{A}(\Pi) \cup \mathcal{P}(\Delta))$, with $\kappa = \beta$ or $\kappa = \mathcal{A}$. If goal $\ell \in \mathsf{P}(\beta)$ is to be enforced by an argument, this is encoded as a *support link* of Π, in a set denoted $\mathcal{SL}(\Pi)$: $(\mathcal{B}, \ell, \beta) \in \mathcal{SL}(\Pi) \subseteq \mathcal{P}(\Delta) \times G(\Pi) \times \mathsf{A}(\Pi)$. (Note an argument $\mathcal{B}$ cannot support some other argument $\mathcal{A}$ *as a link* in $\mathcal{SL}(\Pi)$. To get $\mathcal{B}$ to support step $\mathcal{A}$, just replace step $\mathcal{A}$ by $\mathcal{A} \cup \mathcal{B}$.) Additional *ordering constraints* between action steps are encoded simply as $(\alpha, \beta) \in \mathcal{OC}(\Pi) \subseteq \mathsf{A}(\Pi) \times \mathsf{A}(\Pi)$. The union of causal links, support links (ignoring their $G(\Pi)$ component) and ordering constraints $\mathcal{OC}(\Pi)$ induce, by taking the transitive

closure, the *partial order of* Π, i.e. the order between its steps, denoted $\prec_\Pi$:

$$\prec_\Pi = \mathsf{tc}(\mathcal{OC}(\Pi) \cup \pi_{\widehat{1}}(\mathcal{CL}(\Pi)) \cup \pi_{\widehat{1}}(\mathcal{SL}(\Pi)))$$

Now we define a DeLP-POP plan Π for $\mathbb{M} = ((\Psi, \Delta), \mathsf{A}, G)$ as a tuple $\Pi = (\mathsf{A}(\Pi), G(\Pi), \mathcal{OC}(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi))$ containing actions to be used $\mathsf{A}(\Pi) \subseteq \mathsf{A}$, current open goals of Π, and links or constraints on the execution ordering.

In DeLP-POP an agent with planning domain $\mathbb{M}$ builds a plan incrementally: she keeps refining it with a new step at a time until a solution (a plan with no unsolved flaws) is found. The algorithm used in (23) is the following: For a given $((\Psi, \Delta), \mathsf{A}, G)$, plan search starts with the empty plan $\Pi_\emptyset$, only containing dummy actions $\alpha_\Psi \prec_\Pi \alpha_G$. At each iteration, with current plan $\Pi_\emptyset(\xi_0, \ldots, \xi_k)$, the algorithm nondeterministically selects an unsolved flaw (a threat, preferably) and a refinement step ξ_{k+1} for it (action-, argument- or threat resolution step); after this refinement we obtain plan $\Pi_\emptyset(\xi_0, \ldots, \xi_k, \xi_{k+1})$, and the algorithm updates the set of detected unsolved flaws, so goals and threats are added (if new) or deleted (if solved). If a failure occurs (no refinement is available), the algorithms backtracks to the parent node.

We will denote by $\mathsf{Plans}(\mathbb{M})$ the graph whose nodes are plans for $\mathbb{M}$, related by *is 1-step refinable into*; the set of solution plans will be denoted by $\mathsf{Sol}(\mathsf{Plans}(\mathbb{M}))$.

2.4.2.3 A DeLP-POP extension for the qualification problem

The new extension is motivated by the existence of two kinds of actions' effects in a planning domain:

- *Strict effects*: these that are not susceptible to fail during the execution; i.e. there is no change in context capable of causing a failure during the execution of this action.
- *Defeasible effects*: these that might be susceptible to fail during the execution if there is a environmental condition that prevents it.

As we will see, only the defeasible effects are likely to be discussed in the argumentative dialogues. Unlike strict effects, defeasible effects are related to the qualification problem (59), which is an important problem currently not supported in many planning architectures. It is concerned with the impossibility of listing all the preconditions required for a real-world action to have its intended effect. For instance, let α (e.g. *"flying*

from Bejing to Taipei") be an action with n effects $\{e_0, e_1, \ldots\} \subseteq \mathsf{Lit}$ (e.g. e_0 ="*be at Taipei city"*), which are defeated by the defeasible conditions $\{d_0, d_1, \ldots\} \subseteq \mathsf{Lit}$ (e.g. d_0 = *"Volcanic ash cloud between Bejing to Taipei"*, d_1 ="*Airport strike in Bejing"*) respectively. Note that, if these defeasible conditions occur, the expected effects of α would not be achieved. The typical solution to this issue is to introduce these defeasible conditions as negated preconditions of α ($\{\overline{d_0}, \overline{d_1}, \ldots\} \subseteq \mathsf{P}(\alpha)$), which must be derived by arguments.

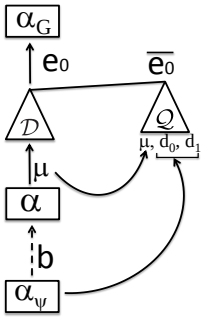

Figure 2.17: An example solving the qualification problem.

However, in this new DeLP-POP extension, an action α (with defeasible effects) follows a specific representation in order to deal with this problem. We introduce a **fictitious effect** μ (meaning α *was just executed*); then we define $\mathsf{X}(\alpha) = \{\mu\}$ and expand the set of rules Δ with $\{e_k \prec\!\!\!- \mu\} \cup \{\overline{e_k} \prec\!\!\!- \mu, d_k\}$, where e_k represents the effect of the action α and d_k is a defeasible condition. For instance, in Figure 2.43, the precondition e_0 of the action α_G is initially derived by an argument $\mathcal{D} = (\{e_0\}, \{e_0 \prec\!\!\!- \mu\})$ whose $\mathsf{base}(\mathcal{D}) = \mu$ will be satisfied by α. Then an attacking argument $\mathcal{Q} = (\{\overline{e_0}\}, \{\overline{e_0} \prec\!\!\!- \mu, d_0, d_1\})$, which is a defeater of $\mathcal{D}$ ($\mathcal{Q}$ attacks $\mathcal{D}$), arises from the distributed knowledge among agents. Triangles in Figure 2.43 represent argument steps (i.e. arguments that support preconditions of action steps), for instance the argument $\mathcal{D}$, or arguments attacking some other argument, for instance the argument $\mathcal{Q}$, and both are labeled with the argument name, while rectangles represent action steps (i.e. actions that support the basis of an argument step) and are labeled with the action name.

Threat detection is based on *proto-states*, defined next. For a fixed $\mathbb{M} = ((\Psi, \Delta), \mathsf{A}, G)$, a plan Π and $\alpha \in \mathsf{A}(\Pi)$, S^{Π}_{α} denotes the set of literals obtaining before α when we extend

$\prec_\Pi$ with some new constraint[1]:

$$S^\Pi_\alpha = \quad \{\ell \in \mathsf{Lit} : \exists \alpha' \in \pi(\ell \in Cn(\{\mu_{\alpha'}\} \cup \Delta) \text{ and } \prec_\pi \cup \{\langle \alpha', \alpha\rangle\} \text{ is consistent,} \\ \text{and } \forall \beta \in \pi(\sim\ell \in Cn(\{\mu_\beta\} \cup \Delta)) \Rightarrow \{\langle \alpha', \beta\rangle, \langle \beta, \alpha\rangle\} \nsubseteq \prec_\pi \cup \{\langle \alpha', \alpha\rangle\})\}$$

Note that, the *proto-state* notion has been slightly changed with respect to the work in (26) to deal with the solution adopted to the qualification problem. We use proto-state S^Π_α to compute which actions or unintended arguments might be triggered by Π in a way interfering with other steps of Π.

Three kinds of threats must be checked during plan construction in DeLP-POP, see also Figure 2.18:

(a) action-action: $(\beta, (\alpha_0, \ell, \alpha_1)) \in \mathsf{A}(\Pi) \times \mathcal{CL}(\Pi)$, s.t. $\overline{\ell} \in \mathsf{X}(\beta)$ and $\prec\ \Pi \cup \{\langle \alpha_0, \beta\rangle, \langle \beta, \alpha_1\rangle\}$ is consistent; here β threatens the link between α_0 and α_1,

(b) action-argument: $((\beta, \overline{n}), (\mathcal{B}, b, \alpha_1)) \in (\mathsf{A}(\Pi) \times \mathsf{Lit}) \times \mathcal{SL}(\Pi)$, with $\overline{\mathsf{X}(\beta)} \cap \mathsf{literals}(\mathcal{B}) \supseteq \{n\}$, where $\prec_\Pi$ makes β to supply $\overline{n} \in S^\Pi_{\alpha_1}$; here β threatens some literal used in $\mathcal{B}$, and

(c) argument-argument: $(\mathcal{C}, (\mathcal{B}, b, \alpha_1)) \in \mathcal{P}(\Delta) \times \mathcal{SL}(\Pi)$, with $\mathcal{C}$ defeating $\mathcal{B}$ and $\mathsf{base}(\mathcal{C}) \subseteq S^\Pi_{\alpha_1}$, $\mathcal{C}$ undefeated in $S^\Pi_{\alpha_1}$.

For each kind of threat, different maneuvers, inspired by those in POP, may be tried: moving the cause of the threat to a harmless position (with new ordering constraints; see Figures 2.19 and 2.20(c')); or eliminating the threat itself (with a counter-argument or a new action step; see Figures 2.20(c")-(c"'). Note a new precondition $\overline{p}$ and new link of type $\mathcal{SL}(\Pi)$ or $\mathcal{CL}(\Pi)$ are needed to preserve these maneuvers' effect in future refinements. Informally, we might see this threat detection-resolution process as generating a dialectical tree $\mathcal{T}_{(S^\Pi_\alpha, \Delta)}(\mathcal{A}_0)$ for each $(\mathcal{A}_0, \cdot, \alpha) \in \mathcal{SL}(\Pi)$. But now the tree is built w.r.t. varying Π, due to new threat resolution refinements. We refer the reader to the algorithm in (23) for details.

2.4.3 Argumentative Dialogues on Multi-agent Plans

The purpose of multi-agent argumentative dialogues is to let agents reach an agreement on (i) the evaluation of plans (Section 2.4.4.1); and (ii) adoption of a plan in decentralized

[1]Note that S^Π_α is computed as if α was already applicable. In particular, arguments occurring before α play no role in S^Π_α.

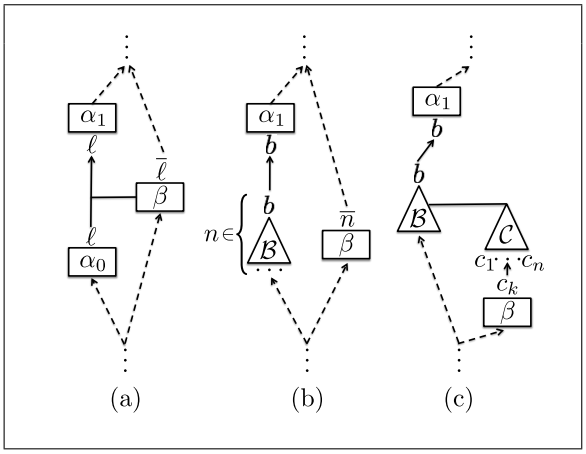

Figure 2.18: Threat types: (a) action-action, (b) action-argument and (c) argument-argument.

plan search (Section 2.4.4.2), by allowing agents to refine or revise other agents' plans and defend one's proposals. Before addressing (i) and (ii), though, several modifications of single-agent DeLP-POP are in order.

First, each agent $x \in \mathsf{Ag}$ is initially endowed with a planning domain $\mathbb{M}_x = ((\Psi_x, \Delta_x), \mathsf{A}_x, G_x)$. Communication (of facts, rules, actions) from agent x to an agent y will be rendered as an expansion (resp., in $\Psi_y, \Delta_y, \mathsf{A}_y$) of $\mathbb{M}_y$.

Second, towards collaborative discovery of potential argument steps or threats and their applicability, agents must send each other known initial facts and pre-arguments; these are like arguments but with partial knowledge of its base, and can be expanded with others' known rules and facts. Given an agent x's plan Π and some $\alpha \in \mathsf{A}(\Pi)$, we define a *pre-argument* $\mathcal{A}$ as a pair of literals and rules $(X, \mathcal{A})$, where $X \subseteq \mathsf{base}(\mathcal{A})$ are literals known to hold before α, and $\mathsf{base}(\mathcal{A}) \setminus X$ contains literals that may not be known that hold, or how to derive them. We define the set of pre-arguments in a proto-state S^{Π}_{α} as $\mathsf{PArgs}(S^{\Pi}_{\alpha}, \Delta_x) := \{(X, \mathcal{A}) \mid X \subseteq S^{\Pi}_{\alpha}, \mathcal{A} \subseteq \Delta_x\}$. Third, we introduce the *cost* of an action, e.g. define action α as $\langle \mathsf{P}(\alpha), \mathsf{X}(\alpha), \mathsf{cost}(\alpha) \rangle$ where $\mathsf{cost}(\alpha) \in \mathbb{R}^+$. This induces an additive plan cost function $\mathsf{cost}(\Pi_{\emptyset}(\xi_0, \ldots, \xi_k) = \Sigma_{k' \leq k} \mathsf{cost}(\xi_{k'})$ that will guide plan search. Another modification needed is the following.

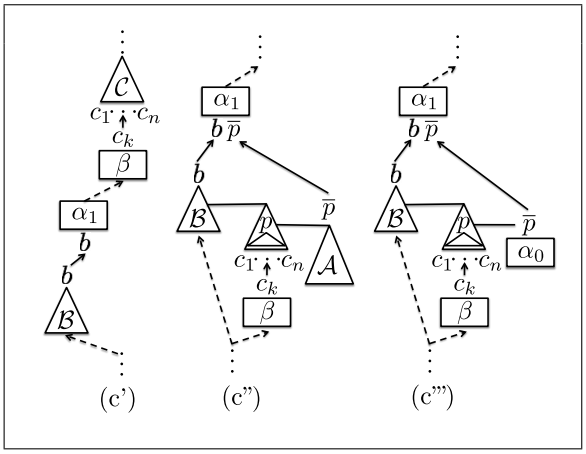

Figure 2.19: Solutions to (a), (b). Demote: (a'), (b'); and Promote: (a"), (b").

Relativizing plans to domains. Even if any plan Π originates from a fixed planning domain $\mathbb{M}$, we can think of so-originated Π also as a plan for some other planning domain $\mathbb{M}'$, and (re-)evaluate Π w.r.t. $\mathbb{M}'$. This is useful when an agent revises her beliefs or is communicated a plan. We denote by $\mathbb{M} \sqsubseteq \mathbb{M}'$ that $\mathbb{M}'$ is an expansion of $\mathbb{M}$, i.e. $\mathbb{M}'$ is such that for all $X \in \mathbb{M}$, its counterpart $X' \in \mathbb{M}'$ satisfies $X \subseteq X'$. And similarly for $T \sqsubseteq T'$. All these expansions may actually translate Π into $\Pi' = \Pi\binom{\alpha_\Psi \alpha_G}{\alpha_{\Psi'} \alpha_{G'}}$.

Lemma 1. *Proto-states S^Π_α are $\subseteq$-monotonic under expansions of T: $T \sqsubseteq T'$ implies $S^\Pi_\alpha \subseteq S^{\Pi'}_\alpha$, where $\Pi' := \Pi\binom{\alpha_\Psi}{\alpha_{\Psi'}}$.*

Also, note that $\mathsf{PArgs}(S^\Pi_\alpha, \cdot)$ is $\subseteq$-monotonic under expansions of Δ: $\Delta \subseteq \Delta'$ makes $\mathsf{PArgs}(S^\Pi_\alpha, \Delta) \subseteq \mathsf{PArgs}(S^\Pi_\alpha, \Delta')$.

Lemma 2. *Action-action and action-argument threats (with action $\neq \alpha_\Psi$) do not increase after expansions of T.*

In contrast, new (α_Ψ) action- and argument-argument threats may appear after expansions of Ψ and, resp., Ψ-or-Δ.

For expansions $\mathbb{M}' \sqsupseteq \mathbb{M}$ a sufficient condition for $\mathbb{M}'$ to accept Π' is that $\mathbb{M}'$ at least contains the elements of Π (and, for Ψ', no more than Ψ).

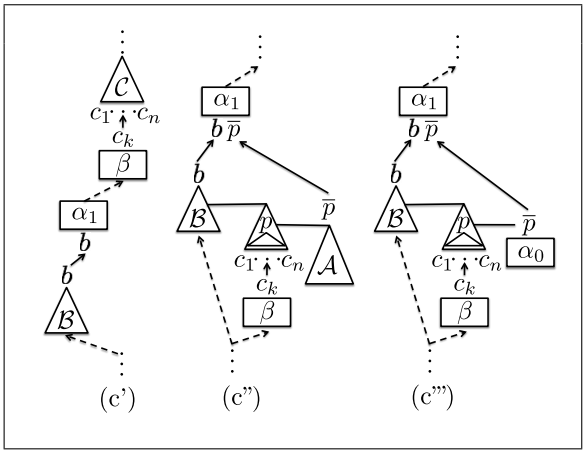

Figure 2.20: Solutions to (c): Delay (c'), Defeat (c") and Disable (c"').

Lemma 3. *Let* $\mathbb{M} = ((\Psi, \Delta), \mathsf{A}, G)$ *be a planning domain and* Π *a plan for* $\mathbb{M}$*. Define* $\mathbb{M}^{\Pi} = ((\Psi^{\star}, \Delta^{\star}), \mathsf{A}^{\star}, G^{\star})$ *as:*
$\Psi^{\star} = \{\ell \in \mathsf{Lit} \mid (\alpha_{\Psi}, \ell, \cdot) \in \mathcal{CL}(\Pi)\}$, $\Delta^{\star} = \bigcup \pi_0[\mathcal{SL}(\Pi)]$, $G^{\star} = G \setminus G(\Pi)$ *and* $\mathsf{A}^{\star} = (\mathsf{A}(\Pi) \setminus \{\alpha_{\Psi}, \alpha_G\}) \cup \{\alpha_{\Psi^{\star}}, \alpha_{G^{\star}}\}$*. Then, for any* $\mathbb{M}' = ((\Psi', \Delta'), \mathsf{A}', G')$ *with* $\Psi' \subseteq \Psi$*,*

$$\Pi\tbinom{\alpha_{\Psi}\alpha_G}{\alpha_{\Psi'}\alpha_{G'}} \textit{ is a plan for } \mathbb{M}' \textit{ iff } \mathbb{M}^{\Pi} \sqsubseteq \mathbb{M}'$$

Only these types of threats that may increase after expansions will be open to argumentation when evaluating the plan's flaws (or its planhood). These results justify the sufficiency of the next relativizations:

Definition 2. *Let* Π *be a* POP *for a given* $((\Psi, \Delta), \mathsf{A}, G)$*, and let* $T' = (\Psi', \Delta')$ *be another de.l.p.. We define the* relativization *of* S^{Π}_{α} *to* Ψ'*,as* $S^{\Psi'}_{\alpha} = S^{\Pi'}_{\alpha}$*, with* $\Pi' = \Pi\tbinom{\alpha_{\psi}}{\alpha_{\psi'}}$*. We denote by* $\mathsf{Threats}^{T'}(\Pi)$ *the set of threats to argument steps in* Π *according to* T'*, as the set of tuples* $(\kappa, (\mathcal{A}, g, \alpha)) \in (\mathcal{P}(\Delta) \cup \mathsf{Lit}) \times \mathcal{SL}(\Pi)$ *such that either:*

$\kappa \subseteq \Delta$, $\mathsf{base}(\kappa) \subseteq S^{\Psi'}_{\alpha}$*,* κ *defeats* $\mathcal{A}$*, and undefeated in* $S^{\Psi'}_{\alpha}$*; or* $\kappa = \ell$*, with* $\ell \in \mathsf{X}(\alpha_{\Psi'}) \cap \overline{\mathsf{literals}(\mathcal{A})}$*, and* $\alpha_{\Psi'}$ *makes* $\ell \in S^{\Psi'}_{\alpha}$ *true.*

Initial dummy action α_{Ψ} is also initially different to each agent. We will assume each agent x, when speaking, uses the convention of referring to *her* initial action, i.e. α_{ψ_x}, by

using the neutral symbol α_Ψ.

2.4.4 Argumentative-Dialogues-based Multi-Agent Planning

In the following, we assume we have a set of agents $\mathsf{Ag} = \{1, \ldots, k\}$, each one with a planing domain $\mathbb{M}_x = ((\Psi_x, \Delta_x), \mathsf{A}_x, G_x)$. In purely cooperative scenarios, agents have no individual interests (i.e. $G_i = G_j$ for any $i, j \in \mathsf{Ag}$) and hence no incentives to retain relevant information. Moreover, we assume $\bigcup_{i \in \mathsf{Ag}} \Psi_i$ is a consistent set. Also, a unique team dialogue to find a solution would suffice. Before presenting dialogues for cooperative plan search, we introduce first a simpler dialogue to evaluate a fixed plan.

2.4.4.1 Argumentative Plan Evaluation

We present now a turn-based dialogue (an agent talking only during her turns) permitting agents i, j to collaborate to discover threats to any argument step $\mathcal{A}$, i.e. with $(\mathcal{A}, \cdot, \alpha) \in \mathcal{SL}(\Pi)$. Here Π is a plan for some $\mathbb{M}_i$ made public. (That is, we assume $\mathbb{M}_\Pi \sqsubseteq \mathbb{M}_x$, $x \in \mathsf{Ag}$.). All agents may contribute to argue against $\mathcal{A}$.

Agents are enumerated by function $\varepsilon : \mathbf{N}^+ \to \mathsf{Ag}$ as: $\varepsilon(i + r \cdot |\mathsf{Ag}|) = i$ for any $r, i \in \mathbb{N}^+$ and $i \leq |\mathsf{Ag}|$; that is, ε assigns turns to agents this way: $1, 2, \ldots, k, 1, 2, \ldots$ At each turn $n + 1$, agent $\varepsilon(n + 1)$ sends a set $\mathbb{A}^{n+1}$ of pre-arguments[1] $(X, \mathcal{B})$ or initial facts $(\emptyset, \ell)$, against an argument $\mathcal{A}$ used in some support link $(\mathcal{A}, \cdot, \alpha)$. For each $(X, \mathcal{B}) \in \mathbb{A}^{n+1}$, any other agent $j \neq \varepsilon(n + 1)$ learns as initial facts those literals stated in X that are not in her view of the proto-state, i.e. with $\ell \in X \setminus S_\alpha^{\psi_j^n}$. All rules from $\mathcal{B}$ which are novel to j are learned as well. Formally,

Definition 3. *For $x \in \mathsf{Ag}$ let $\mathbb{M}_x = ((\Psi_x, \Delta_x), \mathsf{A}_x, G)$ be given, and $\varepsilon : \mathbb{N}^+ \to \mathsf{Ag}$ as above. Let Π be a plan communicated by, say, agent 1 to Ag. We define for each $x \in \mathsf{Ag}$, $\mathbb{A}^0 = \emptyset$, $\psi_x^0 = \Psi_x$, $\Delta_x^0 = \Delta_x$ and*

[1] By exchanging arguments only, an agent might fail to share information, if unaware of its relevance.

$$\begin{aligned}
\mathbb{A}^{n+1} = \{&(\kappa, (\mathcal{A}, \cdot, \alpha) \in \mathcal{P}(\mathsf{Lit}) \times \mathsf{Threats}^{T^{n+1}_{\varepsilon(n+1)}}(\Pi) \mid \\
&\textit{either } \kappa = (X, \mathcal{B}) \textit{ and } X \subseteq \mathsf{base}(\mathcal{B}) \cap S_\alpha^{\psi^{n+1}_{\varepsilon(n+1)}}; \\
&\textit{or } \ \kappa = (\emptyset, \ell) \in \{\emptyset\} \times \mathsf{X}(\alpha_{\psi^{n+1}_{\varepsilon(n+1)}})\} \\
\psi^{n+1}_x = \psi^n_x \cup \bigcup\{&X \setminus S_\alpha^{\psi^n_x} \mid ((X, \mathcal{B}), (\mathcal{A}, \cdot, \alpha)) \in \mathbb{A}^{n+1}\} \\
&\cup\{\ell \in \mathsf{Lit} \mid ((\emptyset, \ell), (\mathcal{A}, \cdot, \cdot)) \in \mathbb{A}^{n+1}\} \\
\Delta^{n+1}_x = \Delta^n_x \cup (&\pi_1[\mathbb{A}^{n+1}] \setminus \mathsf{Lit})
\end{aligned}$$

Finally, let $n^\star$ be the smallest number such that $\mathbb{A}^{n^\star} = \ldots = \mathbb{A}^{n^\star+|\mathsf{Ag}|} = \emptyset$. We define $\psi^\omega_x = \psi^{n^*}_x$, and $\Delta^\omega_x = \Delta^{n^*}_x$.

First note that literals learned in ψ^{n+1}_x from some $((X, \mathcal{B}), \cdot) \in \mathbb{A}^{n+1}$ really come from the agent $n+1$'s ψ-set and propagated to this proto-state.

Lemma 4. *If $\ell \in X \setminus S_\alpha^{\psi^n_x}$ for some $((X, \mathcal{B}), (\mathcal{A}, \cdot, \alpha)) \in \mathbb{A}^{n+1}$, then $\ell \in \psi^n_{\varepsilon(n+1)}$.*

Also note that, since the de.l.p. of each agent is finite, $n^\star$ is finite, i.e. these dialogues will always terminate in a finite number of steps. This dialogue is compared next with centralized plan evaluation, where (a) we consider the *fusion* of agents' initial de.l.p.'s $T_{\Sigma\mathsf{Ag}} = (\Psi_{\Sigma\mathsf{Ag}}, \Delta_{\Sigma\mathsf{Ag}}) = (\bigcup_{x\in\mathsf{Ag}} \Psi_x, \bigcup_{x\in\mathsf{Ag}} \Delta_x)$, and then (b) a central planner computes arguments and threats in this new de.l.p. $(\Psi_{\Sigma\mathsf{Ag}}, \Delta_{\Sigma\mathsf{Ag}})$. The next theorem, then, compares the result of any agent after the evaluation dialogue for $\mathsf{Threats}^{T^\omega_x}(\Pi)$ with that of centralized evaluation $\mathsf{Threats}^{T_{\Sigma\mathsf{Ag}}}(\Pi)$. Even if $T^\omega_x \sqsubset T_{\Sigma\mathsf{Ag}}$ may hold, both evaluations agree on threats detected in Π and whether Π is a plan.

Theorem 1. *Given $\mathbb{M}_x = ((\Psi_x, \Delta_x), \mathsf{A}, G)$ for each $x \in \mathsf{Ag}$, Π a plan for $\mathbb{M}_1$ communicated to $\mathsf{Ag} \setminus \{1\}$. Then, for each x, Π is a plan for $((\psi^\omega_x, \Delta^\omega_x), \mathsf{A}, G)$ iff it is for $(T_{\Sigma\mathsf{Ag}}, \mathsf{A}, G)$, and $\mathsf{Threats}^{(\psi^\omega_x, \Delta^\omega_x)}(\Pi) = \mathsf{Threats}^{(\Psi_{\Sigma\mathsf{Ag}}, \Delta_{\Sigma\mathsf{Ag}})}(\Pi)$*

2.4.4.2 Dialogue-based A^* plan search

The next step is to use these dialogues as part of more dynamic dialogues wherein new plans are proposed. The main result of this paper is that we can decentralize multi-agent planning, at least in cooperative scenarios, by using a dialogue-based plan search procedure. This is done by comparing these dialogues with centralized planning in the fusion of agents' planning domains $\mathbb{M}_{\Sigma\mathsf{Ag}} = (T_{\Sigma\mathsf{Ag}}, \mathsf{A}_{\Sigma\mathsf{Ag}}, G)$, where $\mathsf{A}_{\Sigma\mathsf{Ag}} = \bigcup_{x\in\mathsf{Ag}} \mathsf{A}_x$. But first, we recall A^* search and show it can be used in single-agent DeLP-POP.

A^* search in DeLP-POP. Search algorithms, in the literature, are abstractly defined with non-deterministic choice. In DeLP-POP plan search we saw two such places for non-deterministic choice exist: the selection of the next flaw to be solved (this is optional) and a selection function g for the next refinement, based on minimizing some evaluation function $f(\Pi)$ that estimates the cost of a solution refining Π. As examples of such heuristics to flaw selection: FAF, where flaws are according *fewer alternatives first*, as (60)'s Z-LIFO. Or the threat detect-&-solve order used in (23)'s algorithm.

We opt for an A^* search algorithm, based on delayed termination and an additive evaluation function $f(\Pi) = \mathsf{cost}(\Pi) + f'(\Pi)$, where $f'(\Pi)$ is some heuristic estimation of the cost of some best solution $\Pi^\star$ extending Π.

Recall that A^* procedure is as follows. Start with the initial node $\Pi_\emptyset$, and define sets $\mathsf{open} = \{\Pi_\emptyset\}$ and $\mathsf{closed} = \emptyset$. At each iteration, open is expanded with all generated refinements of current node Π, while Π is sent to closed. Then, we minimize $f[\mathsf{open}]$ to select a refinement $\Pi(\xi)$.

Notice that A^* does not terminate at the first solution, but keeps exploring for less costly possibilities, guided by $g(\mathsf{open}) = \mathrm{argmin}(f(\mathsf{open}))$. If, moreover, f' is optimistic, i.e. $f'(\Pi) \leq f'(\Pi^\star) = \mathsf{cost}(\Pi^\star$, then this $A^\star$ search finds an optimal solution (if a solution exists). Below we will consider the particular case $f'(\Pi) = 0$, so our next-refinement choice function will be just $g(\mathsf{open}) := \mathrm{argmin}(\mathsf{cost}[\mathsf{open}])$.

For a given planning domain $\mathbb{M}$, we define $\mathsf{Plans}_g(\mathbb{M})$ as the set of nodes in $\mathsf{Plans}(\mathbb{M})$ that are generated under A^* search with g.

Proposition 1. *If f' be optimistic, g is admissible for* DeLP-POP *search:* $\mathsf{Sol}(\mathsf{Plans}(\mathbb{M})) \neq \emptyset$ *iff* $\mathsf{Sol}(\mathsf{Plans}_g(\mathbb{M})) \neq \emptyset$*, and a solution $\Pi^\star$ in the latter is optimal.*

The reason is as follows. Suppose $\mathbb{M} = (T, \mathsf{A}, G)$ is a finite domain, so that the cost of any action $\alpha \in \mathsf{A}$ has positive lower bound $\mathsf{cost}[\mathsf{A}] \geq \delta > 0$. Then if (T, A, G) is solvable, a search algorithm guided by g is guaranteed to output an optimal solution in $\mathsf{Sol}(\mathsf{Plans}_g(\mathbb{M}))$ if every infinite path has unbounded cost (see (67)). To see this: if the path contains infinite action steps then it is unbounded, since A is finite implies that $0 < \delta \leq \mathsf{cost}[\mathsf{A}]$ for some δ. Now, if $\mathbb{M}$ is finite, so is $\mathsf{flaws}(\Pi)$; hence null-cost threat resolution moves must be finite. The same reasoning, plus the no-argument-supports-argument policy, implies there can be no infinite sequence of null cost argument steps so we are done.

Hence, A^* can be applied to DeLP-POP plan search for a fixed domain, e.g. centralized $\mathbb{M}_{\Sigma\mathsf{Ag}}$. Below, we show that A^* is also applicable to *dialogue-based* multi-agent plan search.

A^* search in cooperative DeLP-POP. Given agents $\mathsf{Ag} = \{1, \ldots, k\}$, decentralized plan search is also realized as a turn-based dialogue. Turns are now of the form $(n, m) \in \mathbb{N} \times \mathbb{N}$, ordered lexicographically: (n, m) occurs before (n', m') iff $n < n'$, or $n = n'$ and $m < m'$. The agent speaking at (n, m) is $\varepsilon(m)$, who sends a set $\mathbf{\Pi}^{(n,m)}$ of refinements of the plan selected at the n-th iteration of A^*, and a set $\mathcal{U}^{(n,m)}$ of potential threats to previous plans in $\mathbf{\Pi}^{(n,m')}$ for $m' \leq m$. Potential threats are now labeled with the link *and* the plan targeted, say Π' in $\mathbf{\Pi}^{(n,m')}$. In terms of evaluation dialogues, $\mathcal{U}^{(n,m)}$ contains, for each such Π', the corresponding $\mathbb{A}^{m-m'} \times \{\Pi'\}$ (under some permutation $\tau : \mathsf{Ag} \to \mathsf{Ag}$ and initial domains set at $\langle \mathbb{M}^{(n,m')}_{\tau(x)} \rangle_{x \in \mathsf{Ag}}$).

Other agents $x \neq \varepsilon(m)$ learn from $\mathcal{U}^{(n,m)}$ and $\mathbf{\Pi}^{(n,m)}$: (1) literals from pre-arguments and causal links of the form $(\alpha_\Psi, \ell, \cdot)$, (2) rules from pre-arguments and support links, and (3) other agents' actions from suggested plans. This grants that each $\Pi' \in \mathbf{\Pi}^{(n,m)}$ is understood: $\mathbb{M}^{\Pi'} \sqsubseteq \mathbb{M}^{(n,m)}_x$.

Only when, during $|\mathsf{Ag}|$ successive turns $(n, m), \ldots, (n, m + |\mathsf{Ag}|)$, agents do not submit more plans or possible threats, we set $\omega(n) = m$ and move to turn $(n + 1, 0)$. To do so, the set of open nodes is updated with refinements for the current plan: $\mathbf{\Pi}^{(n,\omega(n))} = \mathbf{\Pi}^{(n-1,\omega(n-1))} \cup \bigcup_m \mathbf{\Pi}^{(n,m)}$.

At $(n + 1, 0)$ agents select the best of open nodes: $\mathbf{\Pi}^{(n+1,0)} = \{g(\mathbf{\Pi}^{(n,\omega(n))}\}$. If this contains no flaw, the dialogue terminates. Otherwise the procedure starts again for this plan.

Definition 4. *Given $\mathbb{M}_x = ((\psi_x, \Delta_x), \mathsf{A}_x, G)$ as before, we set $\mathbb{M}^{(0,0)}_x := \mathbb{M}_x$ and define $\mathbf{\Pi}^{(0,0)} = \mathcal{U}^{(0,\cdot)} = \mathcal{U}^{(\cdot,0)} = \emptyset$, $\mathsf{flaw}(0) = h(G)$, and $\mathbf{\Pi}^{(0,1)} = \{\Pi_\emptyset\}$. And,*

$$\begin{aligned}
\mathbf{\Pi}^{(n,m+1)} &= \{\Pi(\xi) \in \mathsf{Plans}(\mathbb{M}^{(n,m)}_{\varepsilon(m+1)}) \mid \Pi \in \mathbf{\Pi}^{(n,0)}, \textit{ and} \\
&\qquad\qquad \mathsf{flaws}(\Pi(\xi)) \smallsetminus \mathsf{flaws}(\Pi) \neq \emptyset\} \\
\mathbf{\Pi}^{(n+1,\omega(n+1))} &= (\mathbf{\Pi}^{(n,\omega(n))} \smallsetminus g(\mathbf{\Pi}^{(n,\omega(n))})) \cup \mathbf{\Pi}^{(n,m_n)}, \\
&\textit{where } m_n = \textit{min } m \textit{ s.t. } \mathbf{\Pi}^{(n,m)} = \ldots = \mathbf{\Pi}^{(n,m+|\mathsf{Ag}|-1)} \\
&\qquad\qquad \textit{and } \mathcal{U}^{(n,m)} = \ldots = \mathcal{U}^{(n,m+|\mathsf{Ag}|-1)} = \emptyset \\
\mathbf{\Pi}^{(n+1,0)} &= \{g(\mathbf{\Pi}^{(n,\omega(n))})\}
\end{aligned}$$

$$\mathcal{U}^{(n,m+1)} = \{(\kappa_0, \kappa_1), (\kappa', \ell, \kappa''), \Pi') \mid \Pi' \in \mathbf{\Pi}^{(n,m+1)} \textit{ and } (\kappa_1, (\kappa', \ell, \kappa'')) \in \mathsf{Threats}^{T^{(n,m)}_{\varepsilon(m+1)}}(\Pi') \textit{ and } \kappa_0 \subseteq \mathsf{base}(\kappa_1) \textit{ or } (\kappa_0, \kappa_1) \in \{\emptyset\} \times \mathsf{Lit}\}$$

At turns of the form $(n, m+1)$ agents learn as follows:

Definition 5. *Each agent $x \neq \varepsilon(m+1)$ updates, at turn $(n, m+1)$,*

$$\begin{aligned}
\psi_x^{(n,m+1)} &= \psi_x^{(n,m)} \cup (\pi_1(\mathcal{U}^{(n,m+1)}) \cap \mathsf{Lit}) \cup \\
&\quad \bigcup\{X \smallsetminus S_{\alpha_1}^{\psi_x^{(n,m)}} \mid ((X, \mathcal{B}), (\mathcal{A}, \cdot, \alpha_1)) \in \mathcal{U}^{(n,m+1)}\} \\
\Delta_x^{(n,m+1)} &= \Delta_x^{(n,m)} \cup \{\pi_0(\xi) \mid \xi \in \mathcal{SL}[\mathbf{\Pi}^{(n,m+1)}] \cup \ldots \\
&\quad \cup \pi_1(\{(\kappa, \ldots) \in \mathcal{U}^{(n,m+1)}) \mid \pi_1(\kappa, \ldots) \notin \mathsf{Lit}\}) \\
\mathsf{A}_x^{(n,m+1)} &= \mathsf{A}_x^{(n,m)} \cup \{\alpha \in \mathsf{A}_{\Pi(\xi)} \mid \Pi(\xi) \in \mathbf{\Pi}^{(n,m+1)}\}
\end{aligned}$$

For sets $X_x^{(n,\cdot)}$ defined here plus $\mathbb{M}_x^{(n,\cdot)}$ we define $X_x^{(n+1,0)} = X_x^{(n,\omega(n))} = \bigcup_m X_x^{(n,m)}$, and $X_x^{\omega} = \bigcup_{n\in\omega} X_x^{(n,0)}$.

Theorem 2. *Let $\langle \mathbb{M}_x \rangle_{x\in\mathsf{Ag}}$ and g be as above. Then, $\mathsf{Sol}(\mathsf{Plans}_g(\mathbb{M}_{\Sigma\mathsf{Ag}})) \neq \emptyset$ iff $\mathsf{Sol}(\mathsf{Plans}_g(\mathbb{M}_x^{\omega})) \neq \emptyset$, for any x; moreover, a solution $\Pi^\star$ in the latter is optimal.*

Thus, agents may safely use these dialogues to find an optimal, cooperative plan which makes use of their abilities.

2.4.5 A scenario of validation

Figure 2.21 shows a scenario to Multi-Agent Planning. There are three different locations in this scenario $Bejing$, $Fuzhou$ and $Taipei$. Our multi-agent systems is composed of two agents, Joe and Ann, who wish to travel to $Taipei$ to attend the AAMAS conference as invited speakers. As can be seen, there are several direct or indirect connections between $Bejing$ and $Taipei$: via car and ship, train and ship, or plane. The agents, the car, the train and the plane are initially located at $Bejing$, and the goal ($G = \{(at\ \mathsf{Ag}\ l3)\}$) is to have the two agents at $Taipei$ subject to the restriction that they must always travel together. We consider propositional STRIPS planning representation, and the default proposition $(have\ p)$ to any literal p that does not have an associated proposition. Literals and actions are the following:

- $l1, l2, l3$ - $Bejing$, $Fuzhou$ and $Taipei$,

- car, tra, pl, shi - a car, a train, a plane, a ship,
- r, rl, al, ml - a road, a railway, an airline company, a maritime line,
- bw, sn, wg, ss - bad weather, snow, wind gusts, stormy sea,
- br, ll, esf, aeo - bad railroad, landslides, electrical supply failure, airplane engines work well (after test)
- va, ds, ip, gw - volcano ash cloud, dangerous situation, risk of increased pollution, contribution to global warming,
- h, tj, $kudTV$, $kudI$ - holidays, traffic jam, kept up to date by TV news, kept up to date by Internet news,
- μ_C, μ_P, μ_T, μ_S - moved car, moved plane, moved train and moved ship

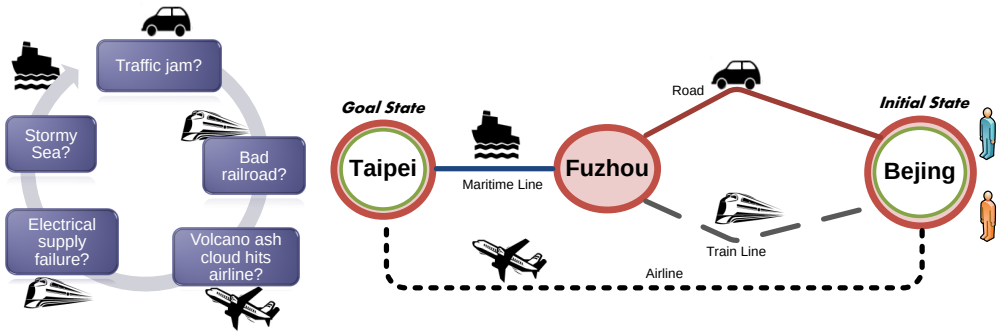

Figure 2.21: Scenario of the application example

1. $mP(pl, j, k)$: moving plane 'pl' from location 'j' to 'k'. It is necessary an airline company to travel from 'j' to 'k', the plane in 'j' and both Joe and Ann in 'j'. Moving a plane takes 2 time unit and 400 cost units.
2. $mT(tra, j, k)$: moving train 'tra' from location 'j' to 'k'. This action takes 6 time units and 200 cost units.
3. $mS(shi, j, k)$: moving ship 'shi' from location 'j' to 'k'. This action takes 3 time units and 100 unit cost.

$$\mathsf{A}^{(0,0)}_{Joe} = \left\{ \begin{array}{l} 1.\ \{\mu_C, ip\} \xleftarrow{fMc} \{(link\ r\ l1\ l2),\ (at\ car\ l1), \\ (at\ \mathsf{Ag}\ l1)\} \\ 2.\ \mu_P \xleftarrow{mP} \{(link\ al\ l1\ l3),\ (at\ pl\ l1), \\ (at\ \mathsf{Ag}\ l1)\} \end{array} \right\}$$

$$\mathsf{A}^{(0,0)}_{Ann} = \left\{ \begin{array}{l} 3.\ \mu_T \xleftarrow{mT} \{(link\ rl\ l1\ l2),\ (at\ tra\ l1), \\ (at\ \mathsf{Ag}\ l1)\} \\ 4.\ \mu_S \xleftarrow{mS} \{(link\ ml\ l2\ l3),\ (at\ shi\ l2), \\ (at\ \mathsf{Ag}\ l2)\} \end{array} \right\}$$

$$\Psi^{(0,0)}_{Joe} = \left\{ \begin{array}{c} wg;\ aeo;\ kudTV; (at\ \mathsf{Ag}\ l1); \\ (at\ pl\ l1);\ (link\ al\ l1\ l3);\ (link\ r\ l1\ l2); \end{array} \right\}$$

$$\Psi^{(0,0)}_{Ann} = \left\{ \begin{array}{c} kudI;\ (at\ \mathsf{Ag}\ l1);\ (at\ tra\ l1);\ (at\ shi\ l2) \\ (link\ rl\ l1\ l2);\ (link\ ml\ l2\ l3) \end{array} \right\}$$

Figure 2.22: Knowledge of actions and initial facts.

4. $fMc(car, j, k)$: fast-moving car 'car' from location 'j' to 'k'. This action takes 8 time units and 80 cost units.

We describe next the initial planning domains: for $x \in \mathsf{Ag} = \{Ann, Joe\}$, let $\mathbb{M}^{(0,0)}_x = ((\Psi^{(0,0)}_x, \Delta^{(0,0)}_x), \mathsf{A}^{(0,0)}_x, G)$ be defined as in Figures 2.22 and 2.23. Actions $\alpha = (\mathsf{P}(\alpha), \mathsf{X}(\alpha), \cdot)$ are represented under the form $\mathsf{X}(\alpha) \xleftarrow{\alpha} \mathsf{P}(\alpha)$. Ann and Joe have different knowledge so two pieces of derived information from each agent can appear to be contradictory. Let's assume that Joe uses TV as a source of information, but Ann prefers Internet to keep up to date, and both agree in finding a plan that minimizes the time units.

In what follows, we explain how to obtain an optimal plan $\Pi^\star$ that satisfies the goal $G = \{(at\ \mathsf{Ag}\ l3)\}$.

The planning process starts with Ann's empty plan $\Pi_\emptyset$, essentially, $\{\alpha_\emptyset \prec \alpha_G\}$ and $\mathcal{U}^{(0,1)} = \emptyset$. Joe learns nothing from it; and both agents set $g(\mathbf{\Pi}^{(0,\omega(0))}) = \Pi_\emptyset$. Then $\mathsf{flaws}(\Pi)$ returns $(at\ \mathsf{Ag}\ l3)$. At turn (1,1) Ann suggests the ship argument, while at next

$$\Delta^{(0,0)}_{Joe} = \left\{ \begin{array}{c} \{(at\ pl\ l3), (at\ \mathsf{Ag}\ l3)\} \multimap \mu_P; \\ \{(at\ car\ l2), (at\ \mathsf{Ag}\ l2)\} \multimap \mu_C; \\ \{\sim(at\ tra\ l2), \sim(at\ \mathsf{Ag}\ l2)\} \multimap \{\mu_T, br\}; \\ \{\sim(at\ shi\ l3), \sim(at\ \mathsf{Ag}\ l3)\} \multimap \{\mu_S, ss\}; \\ br \multimap ll;\ ll \multimap wg;\ br \multimap esf;\ esf \multimap sn; \\ sn \multimap kudTV;\ tj \multimap h;\ h \multimap kudTV; \\ ss \multimap bw;\ bw \multimap wg;\ \sim va \multimap aeo; \end{array} \right\}$$

$$\Delta^{(0,0)}_{Ann} = \left\{ \begin{array}{c} \{\sim(at\ pl\ l3), \sim(at\ \mathsf{Ag}\ l3)\} \multimap \{\mu_P, ds\} \\ \{\sim(at\ car\ l2), \sim(at\ \mathsf{Ag}\ l2)\} \multimap \{\mu_C, tj\} \\ \{(at\ tra\ l2), (at\ \mathsf{Ag}\ l2)\} \multimap \mu_T; \\ \{(at\ shi\ l3), (at\ \mathsf{Ag}\ l3)\} \multimap \mu_S; \\ ds \multimap va;\ va \multimap kudI;\ \sim ss \multimap \sim bw; \\ \sim bw \multimap h;\ h \multimap kudI;\ \sim ll \multimap \sim bw;\ \sim br \multimap \sim bw; \\ \sim bw \multimap kudI;\ \sim sn \multimap kudI;\ gw \multimap ip; \end{array} \right\}$$

Figure 2.23: Defeasible rules known by each agent.

turn $(1, 2)$, Joe puts forward this argument step (Figure 2.24(a)):

$\Pi_\emptyset(\xi^{Joe}) \in \mathbf{\Pi}^{(1,2)}$ where $\xi^{Joe} = (\mathcal{A}^{\text{Joe}}, (at\ \mathsf{Ag}\ l3), \alpha_G))$ and $\mathcal{A}^{\text{Joe}} = (\{(at\ \mathsf{Ag}\ l3) \multimap \mu_P\})$. Ann learns the rule in $\mathcal{A}^{\text{Joe}}$. This is the plan with less cost, so it selected at $\mathbf{\Pi}^{(2,0)}$ with $\mathsf{flaws}(\Pi_\emptyset(\xi^{Joe})) = \{\mu_P\}$.

At $(2, 1)$ turn, Ann cannot refine this plan. This is done, at turn $(2, 2)$ by Joe: $\Pi_\emptyset(\xi^{Joe}, (mP, \mu_P, \mathcal{A}^{\text{Joe}})) \in \mathbf{\Pi}^{(2,2)}$, where he proposes the action $mP(pl,\ l1,\ l3)$ to enforce μ_P (Figure 2.24(b)). Let Π' denote this plan. Each agent x learns in $(2, 2)$ that $\mu_P \in S^{\psi^{(2,2)}_x}_{\alpha_G}$. Ann learns action mP.

Now its Ann's turn $(2, 3)$. She finds an argument-argument threat to $\mathcal{A}^{\text{Joe}}$ based on her initial knowledge of $kudI$. She sends $\mathcal{U}^{(2,3)} = \{((\{kudI\}, \mathcal{B}^{\text{Ann}}), (\mathcal{A}^{\text{Joe}}, at\ \mathsf{Ag}\ l3, \alpha_G), \Pi')\}$ where $\mathcal{B}^{\text{Ann}} = \{\sim(at\ \mathsf{Ag}\ l3) \multimap \{\mu_P, ds\};\ ds \multimap va;\ va \multimap kudI\}$ (Figure 2.24(c)). The initial fact $kudI$ and these rules are learnt by Joe. Assume Joe's plan is selected at $\mathbf{\Pi}^{(3,0)}$ with $\mathsf{flaws}(\Pi')$ containing Ann's threat based on $\mathcal{B}^{Ann}$.

At Ann's turn $(3, 1)$, she finds nothing else relevant to Joe's plan. Joe's turn (3,2). To solve Ann's threat, Joe selects a *Defeat* move against ds, based on his knowledge.

$\mathbf{\Pi}^{(3,2)} = \{\Pi'(\text{Defeat}(\mathcal{C}^{\text{Joe}}, \mathcal{B}^{\text{Ann}}))\}$ where $\mathcal{C}^{\text{Joe}} = (\{aeo\}, \{\sim va \prec aeo\})$. It is a Defeat resolution move since: $\sim \text{concl}(\mathcal{C}^{\text{Joe}}) \in \text{literals}(\mathcal{B}^{\text{Ann}})$) (Figure 2.25(d)).

In summary, Joe suggested to take the plane to arrive to $Taipei$, but Ann attacked the proposal because the volcano ashes are expected according to the Internet information, and Joe replied that this situation will not affect the flight between $Beijing$ and $Taipei$ (according to the results on engine tests). For space reasons, we omit the rest of the dialogue showing this is plan can be refined to an optimal solution.

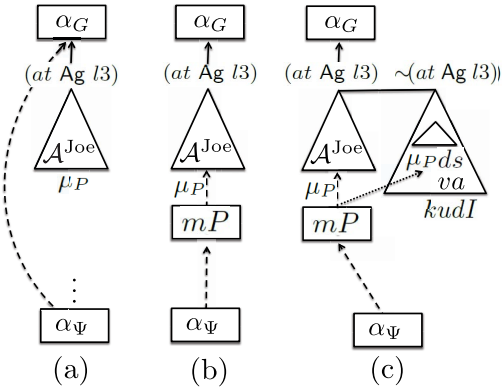

Figure 2.24: (a), (b): Joe's turns and (c): Ann's turn

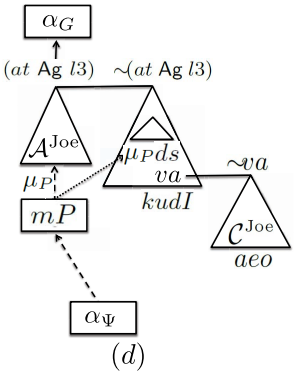

Figure 2.25: (d): Joe's turn

2.4.6 Related Work

The work presented here is similar to several proposals found in the literature: multi-agent argumentation (in non-dynamic scenarios), cooperative planning (without defeasible argumentation) and centralized planning.

Some systems that build on argumentation apply theoretical reasoning for the generation and evaluation of arguments to build applications that deal with incomplete and contradictory information in dynamic domains. Some proposals in this line focus on planning tasks, or also called practical reasoning, i.e. reasoning about what actions are the best to be executed by an agent in a given situation. Dung's abstract system for argumentation (4) has been used for reasoning about conflicting plans and generate consistent sets of goals (6, 51). Further extensions of these works present an explicit separation of the belief arguments and goals arguments and include methods for comparing arguments based on the worth of goals and the cost of resources (5). In any case, none of these works apply to a multi-agent environment.

A proposal for dialogue-based centralized planning is that of (53), but no argumentation is made use of. The work in (29) 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. The works in (68, 69) present an argumentation scheme to propose and justify plans. Unlike our approach of considering defeasible reasoning, these last works are referred to critical questions that address specific elements of the plan proposals. On the other hand, we can also find some systems that realize argumentation in multi-agent systems using defeasible reasoning but are not particularly concerned with the task of planning (41).

All in all, the novelty of our approach is the combination of all these aspects: defeasible reasoning, decentralized planning and multi-agent systems.

2.4.7 Conclusions and Future Work

We have presented a decentralized A^* plan search algorithm for multiagent argumentative planning in the framework of DeLP-POP. This search is implemented as a dialogue between agents, which cooperate to criticize or defend alternative plans by means of defeasible arguments. Only potentially relevant information is exchanged in the dialogue

process, which terminates in a provably optimal solution upon which agents cannot disagree.

As future work, we have the intention of:

- extending the present approach to model Argumentation-Based Negotiation, so that ambient agents are able to negotiate on refinement plans (70, 71).
- extending the present approach to Temporal Planning by using t-DeLP (65, 72); we believe that exploiting temporal argumentation and planning would report important benefits not only as for *what to do* but also *when to do* it in order to achieve a temporally consistent set of goals;
- studying the influence of the trust on the sources used by the planning agents to acquire the defeasible rules as well as how a trust level may determine the conflict resolution between arguments;
- implementing the presented multi-agent framework in an agent platform, which allows us to test this model in various real-world application domains (for instance, Ambient Intelligence applications (27) or Transit Journey Planning Services (34, 73)) by getting experimental results in order to validate it. In this way, we will experimentally test the performance, scalability and quality of multi-agent DeLP-POP versus DeLP-POP, and multi-agent DeLP-POP versus a Multi-Agent Planning system with no argumentation. For more details about this last extension see the work in (27).

Acknowledgments. The authors acknowledge partial support of the Spanish MICINN projects CONSOLIDER-INGENIO 2010 Agreement Technologies CSD2007-00022; TIN2009-14704-C03-03; LoMo-ReVI FFI2008-03126-E/FILO (FP006); FPU grant reference AP2009-1896 awarded to Sergio Pajares Ferrando; TIN2011-27652-C03-01; TIN2008-04446 and PROMETEO/2008/051; and the Generalitat de Catalunya grant 2009-SGR-1434.

2.5 Selected Paper 4: Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications (AAMAS 2012)

Abstract. *This contribution presents a practical extension of a theoretical model for multi-agent planning based upon* DeLP*, an argumentation-based defeasible logic. Our framework, named* DeLP-MAPOP*, is implemented on a platform for open multi-agent systems and has been experimentally tested, among others, in applications of ambient intelligence in the field of health-care.* DeLP-MAPOP *is based on a multi-agent partial order planning paradigm in which agents have diverse abilities, use an argumentation-based defeasible reasoning to support their own beliefs and refute the beliefs of the others according to their knowledge during the plan search process. The requirements of Ambient Intelligence (AmI) environments featured by the imperfect nature of the context information and heterogeneity of the involved agents make defeasible argumentation be an ideal approach to resolve potential conflicts caused by the contradictory information coming from the ambient agents. Moreover, the ability of AmI systems to build a course of action to achieve the user's needs is also a claiming capability in such systems.* DeLP-MAPOP *shows to be an adequate approach to tackle AmI problems as it gathers together in a single framework the ability of planning while it allows agents to put forward arguments that support or argue upon the accuracy, unambiguity and reliability of the context-aware information.*

2.5.1 Introduction

Ambient Intelligence (AmI) integrates concepts ranging from Ubiquitous Computing to Artificial Intelligence with the vision that technology will become invisible, embedded in our natural surroundings, present whenever we need it, and adaptive to users (74). In AmI environments, people are surrounded with networks of embedded intelligent devices that can sense the available context information, anticipate, and perhaps adapt to their needs. In this contribution, we handle these requirements by modeling ambient agents as entities which manage a portion of the AmI environment, i.e. they are responsible for one or more devices. Due to the imperfect nature of the context and the heterogeneity of ambient agents, whose different viewpoints lead them to infer different assumptions about

the user's current situation, ambient agents, as distributed autonomous software entities, are required to engage in interactions, argue with one another, and make agreements, individually or collectively, while responding to changing circumstances of the ambient environment. For this reason, ambient agents are being advocated as a next-generation model for engineering complex distributed systems such as AmI systems. The aim in AmI is to make the interaction between users and the smart environment easy.

Defeasible is the opposite of irrefutable or indisputable. A defeasible piece of information is a non-demonstrative piece of information that is acknowledged to be able to fail or be corrected. Defeasible reasoning is usually realized as a rule-based approach for reasoning with incomplete and inconsistent information through the use of rules that may be defeated by other rules. Defeasible reasoning has been successfully used in AmI applications (63). On the other hand, **Defeasible Argumentation**, which has recently become a very active research field in computer science (75), is a form of defeasible reasoning that emphasizes the notion of argument. An argument is a chain of reasoning that concludes one piece of information (conclusion) on the basis of some other pieces of information (premises). Thus, defeasible argumentation can be viewed as a powerful tool for reasoning about inconsistent information through a rational interaction of arguments for and against some conclusion derived by an ambient agent. Defeasible argumentation has also been successfully proved in AmI applications (76).

The defeasible logic programming formalism DeLP (24) is one of the most popular approaches to build defeasible argumentation. Our framework, DeLP-MAPOP, builds upon DeLP to implement the defeasible argumentation mechanism. The key element of DeLP are defeasible rules (Head $\prec$ Body), which are used to represent a deductive relation between pieces of knowledge that could be defeated once another piece of knowledge is considered. For instance, a defeasible rule like *emergency* $\prec$ *patient-fever* denotes that an ambient agent believes that if the monitoring system returns the patient has fever then there are provable reasons to declare an emergency. The defeasible rule $\sim$*emergency* $\prec$ *{normal-pulse, conscious, correct-breathing}* provides reasons to believe the contrary, in whose case we say that the first piece of information is acknowledged to fail in case *{normal-pulse, conscious, correct-breathing}* hold in the context. However, assuming that another ambient agent knows that the patient is vomiting blood, i.e. {*bloody-vomit*} holds in the context, then it might derive the patient has not a normal pulse by following the defeasible rules {$\sim$*normal-pulse* $\prec$ *internal-bleeding*; *internal-bleeding* $\prec$ *bloody-vomit*},

which represents an attack to the defeasible rule whose conclusion is $\sim$*emergency*. Thus, arguments (combinations of defeasible rules and facts) for conflicting pieces of information are built, and then compared to decide which one prevails.

Planning is a desired ability in AmI systems to achieve a goal-oriented behavior, i.e. to decide the course of action to meet the needs of the specific application, for instance, stabilizing a patient in a home-care system. Planning has been used in some AmI applications for monitoring and responding to the needs of a diabetic patient (77). Particularly, the work in (77) presents a centralized planner that manages distributed capabilities as it assumes that some agents do not have planning capabilities. In this case, an agent is implemented as a device, which prevents the agent from taking responsibilities in building the plan due to its limitations in processing and communication; for example, a cell phone could not be able to autonomously plan to call a doctor given that other devices detected that a user in the environment is ill (77). However, in our contribution an ambient agent is executed on an independent host and can encompass several devices. This increases the communication capacity as well as autonomy and endow agents with the necessary abilities to pose a goal and build a plan for this goal. This approach allows us to address many real applications where the capabilities to perceive the context and perform the actions are distributed across agents. **Multi-Agent Planning** (MAP) applied to an AmI environment is intended as the ability of a team of ambient agents to build collaboratively a plan of actions that, when performed in the AmI context, meets the needs and goals of the application.

Partial Order Planning (POP) is a suitable planning approach to address the requirements derived from a distributed planning thanks to the application of a non-sequential behaviour and the least commitment principle (43). This is evidenced by the fact that most existing architectures for integrating planning with execution, information gathering and scheduling are based on partial order planners. In (54), authors argue that POP-based frameworks offer a more promising approach for handling domains with durative actions and temporal and resource constraints as compared to other planning approaches. In fact, most of the known implementations of planning systems capable of handling temporal and durative constraints (e.g. NASA's RAX (56)) are based on the POP paradigm. Even for simple planning tasks, partial order planners offer a higher degree of execution flexibility. For these reasons, this work is based on Multi-Agent Partial Order Planning (MAPOP).

A extension of POP with DeLP-style argumentation, denoted as DeLP-POP, was

introduced in (23), where both actions and arguments may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. Unlike actions, arguments are not only introduced to intentionally support some step of a plan, but they are also presented to defeat or defend other supporting arguments in the plan. When actions and arguments are combined in a partial order plan, new types of interferences or threats appear (23) which need to be identified and resolved to obtain valid plans. Finally, the work in (26, 34, 73) proposes an extension of the DeLP-POP to a multi-agent environment. Specifically, it proposes a dialogue for argumentative plan search, by which agents exchange plan proposals and arguments for or against such proposals. To the best of our knowledge, these theoretical works have neither been implemented nor tested on real-world domains such as AmI applications.

This contribution presents DeLP-MAPOP, a system that combines, implements and tests features like multi-agent defeasible argumentation and multi-agent planning in AmI applications. DeLP-MAPOP develops and implements an extended and refined version of the framework presented in (26); DeLP-MAPOP is applied and experimentally tested in an AmI environment, it extends the agents' knowledge bases and the dialogues during the plan search and it offers a new classification of planning interferences. The remainder of this paper is divided as follows. First, we introduce the basic elements of the system; then we present the MAP protocol applied in an AmI scenario to deal with a person suffering from a heart disease. Next, the experiments carried out to validate the present work are described and analyzed. Finally, we conclude and present some directions for future work.

2.5.2 Components of the system

In this section, we provide definitions for the notions of ambient agent, context information, planning task, argument versus action and plan, that will be later used for the definition of the DeLP-MAPOP protocol.

2.5.2.1 Ambient Agents

In DeLP-MAPOP, ambient agents act as planning agents with different beliefs, capabilities and preferences. Thus, we assume the capabilities to perceive the context, perform actions and derive new conclusions are distributed across ambient agents. Agents are

managed and supervised by the Agent Management System (AMS) that is responsible for the following tasks: i) Exercising supervisory control over access to the multi-agent platform; it is responsible for authentication of resident ambient agents and control of registrations. ii) Discovering new user's needs generated directly by the user or indirectly by a smart device, which provides the input to a DeLP-MAPOP process in terms of goals to be reached. iii) When ii) occurs, the AMS agent gathers the ambient agents who will participate in the planning process and will return the action plan to satisfy the user's needs. For instance, a device that monitors the patient's heart's rate may detect the presence of arrhythmias by means of an electrocardiogram, a symptom that might entail a heart attack. In this case, the monitoring system generates the goal *patient-to-be-treated*, and communicates it to the AMS agent.

The knowledge of an ambient agent mainly comprises context information encoded as defeasible rules and initial facts, and context capabilities represented as planning actions.

2.5.2.2 Context information

The representation scheme used by DeLP-MAPOP to model components of the AmI environment is based on a state-variable representation, where variables map to a finite domain of values which represent the problem objects. A state-variable representation is equivalent to a classical planning representation in expressive power and it is also useful in non-classical planning problems as a way to handle numbers, functions and time. In this paper, we will restrict our attention to only non-numeric variables. Since actions change the state of the world and defeasible rules make assumptions about the state of the world, actions and defeasible rules are most naturally modeled as elements that change the values of the state variables. The variable-value pair $\langle v_i, vl_i \rangle$ denotes the value vl_i is assigned to the variable v_i. For instance, the variable-value pair $\langle$ *at-amb, pH* $\rangle$ indicates that the ambulance amb is located at the patient's home pH, that is, the value of the variable denoting the position of the ambulance is the patient's home.

In what follows, we define the set of elements used to represent the agent's context information. (i) the set of objects O that model the elements of the planning domain over which the actions and defeasible rules can act. (ii) the set of state variables V that are used to model the states of the world: each state variable $v_i \in V$ is mapped to a finite domain of mutually exclusive values D_{v_i}, where $\forall v_i \in V$, $D_{v_i} \subseteq O$. (iii) the initial state of the problem Ψ, which is a consistent set of variable-value pairs; a variable with

no assigned value in the initial state is assumed to have an *unknown* value. (iv) the set of defeasible rules Δ, where each rule δ follows the form $\langle \mathsf{head}(\delta) \prec \mathsf{body}(\delta) \rangle$; if the set of variable-value pairs in $\mathsf{body}(\delta)$ is warranted, i.e. if variables have the specified values in the pair, then δ is applicable and for each $\langle v_i, vl_i \rangle$ that appears in the head of the rule, v_i is assigned the value vl_i. (v) A is the set of planning actions $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$, where $\mathsf{P}(\alpha)$ is a set of preconditions encoded as variable-value pairs that must be satisfied in order to apply the effects in $\mathsf{X}(\alpha)$, also encoded as $\langle v_i, vl_i \rangle$.

2.5.2.3 Planning task

Each ambient agent $x \in \{\mathsf{Ag}_1 \ldots \mathsf{Ag}_n\}$ is initially endowed with **a planning task** $\mathbb{M}_x = (O_x, V_x, \Psi_x, \Delta_x, A_x, F_x, G)$ where:

1. O_x is the set of objects known by the agent x.

2. V_x is the set of variables managed by agent x to represent the agent's knowledge about the state of the world.

3. $\Psi_x = \{\langle v_i, vl_i \rangle \mid v_i \in V_x; vl_i \in D_{v_i}\}$ represents the partial view of the initial world state of agent x, i.e. the information that agent x knows about the initial state. We assume $\bigcup_{x \in \{\mathsf{Ag}_1 \ldots \mathsf{Ag}_n\}} \Psi_x$ is a consistent set.

4. Δ_x is a set of defeasible rules known by the agent x.

5. A_x is a set of planning actions known by the agent x.

6. F_x represents a consistent set of the agent-specific preferences $F_x \subseteq \{(a, d) \mid (a \in A_x),\ d \in [0, 100]\}$, where action a is preferred with the estimated interest degree d.

7. G is the set of global goals that represent the needs of a user in an AmI environment. G is expressed as a set of pairs variable-value thus indicating the value each variable is expected to assume in the final state. Unlike the rest of elements, G is known by all of the ambient agents.

2.5.2.4 Arguments versus Actions

As we saw in the Introduction section, and based on the framework presented in (23), both actions and arguments may be used to enforce some task goal in DeLP-MAPOP. As illustrated in Figure 2.43 (a), an **argument** $\mathcal{A}$ for $\langle v_i, vl_i \rangle$ proposed by an ambient agent Ag_1, is denoted as $\mathcal{A}^{\mathsf{Ag}_1} = (\{\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1})\}, \{\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})\})$, where $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \langle v_i, vl_i \rangle$ is the argument conclusion and $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})$ is a subset of defeasible rules such that $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1}) \subseteq \Delta_x$. $\mathcal{A}^{\mathsf{Ag}_1}$ is consistent if there exists a defeasible derivation for $\langle v_i, vl_i \rangle$ from $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1}) \cup \mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})$, where $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ is the argument base, the set of <variable,value> that must be warranted in the agent's context information. The existence of an argument $\mathcal{A}^{\mathsf{Ag}_1}$ does not suffice to warrant its conclusion $\langle v_i, vl_i \rangle$, this depends on the interactions among arguments as we will see in Section 2.6.6.2. We semantically distinguish between **supporting arguments** (also known as argument steps) as the arguments specifically used to support some goal of the plan, and **attacking arguments** (also known as defeaters) which are only introduced to attack some argument step previously introduced in the plan.

The difference between assigning a value to a variable by an argument or by an action is that in the case of a planning action the value is indisputable because it reflects a modification stated in the problem domain modelling; however, the confirmation of a value assigned to a variable by an argument depends on the interaction with other attacking arguments.

2.5.2.5 Plans

In POP, a partial order plan Π is a set of partially ordered actions (denoted by the relation $\prec$) which actually encodes multiple linear plans. More specifically, a **plan** Π is a tuple $\Pi = (A(\Pi), \mathcal{AR}(\Pi), G(\Pi), \mathcal{OC}(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi))$, where $A(\Pi)$ denotes the set of action steps, $\mathcal{AR}(\Pi)$ represents the set of argument steps, $G(\Pi)$ is the task's common goals (the user's needs), $\mathcal{OC}(\Pi)$ is a set of ordering constraints, and $\mathcal{CL}(\Pi)$ and $\mathcal{SL}(\Pi)$ represent the sets of causal and support links, respectively. In POP, Ψ and G are encoded as dummy actions $\{\alpha_\Psi \prec \alpha_G\}$ where α_Ψ is also refereed to as the initial step of the plan and α_G to as the final step of the plan, with $\mathsf{X}(\alpha_\Psi) = \Psi$, $\mathsf{P}(\alpha_G) = G$, and $\mathsf{P}(\alpha_\Psi) = \mathsf{X}(\alpha_G) = \emptyset$.

Let $\langle v_i, vl_i \rangle$ be an open goal in Figure 2.43(b), motivated by some action step $\alpha_G \in A(\Pi)$, i.e. $\langle v_i, vl_i \rangle \in \mathsf{P}(\alpha_G)$; let $\langle v_k, vl_k \rangle$ be another open goal, motivated by some

argument step $\mathcal{A}^{\mathsf{Ag}_1}$, i.e. $\langle v_k, vl_k\rangle \in \mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$. Then, the goal $\langle v_i, vl_i\rangle \in \mathsf{P}(\alpha_G)$ must be supported by an argument, argument $\mathcal{A}^{\mathsf{Ag}_1}$ in Figure 2.43(b), which introduces a **support link** $(\mathcal{A}^{\mathsf{Ag}_1}, \langle v_i, vl_i\rangle, \alpha_G) \in \mathcal{SL}(\Pi)$, where $\mathcal{SL}(\Pi) \subseteq \Delta \times G(\Pi) \times A$. In contrast, the goal $\langle v_k, vl_k\rangle$ must be supported by an action, α_1 in Figure 2.43(b), which introduces a **causal link** $(\alpha_1, \langle v_k, vl_k\rangle, \mathcal{A}^{\mathsf{Ag}_1}) \in \mathcal{CL}(\Pi)$, where $\mathcal{CL}(\Pi) \subseteq A \times G(\Pi) \times \Delta$. Triangles in Figure 2.43(b) represent argument steps (i.e. arguments that support preconditions of action steps), while rectangles represent action steps (i.e. actions that support the basis of an argument step). Therefore, in this approach, goals must always be initially derived by some argument step, and an argument base must be satisfied by another action step (including the initial step). This way, a typical causal link in POP is now replaced by a causal link and a support link. Note this representation allows us to implicitly address *the qualification problem* (59) as every precondition of a planning action is now supported by an argument step rather than directly by an action effect. This way, agents may attack the fulfillment of such precondition if they believe that there exist other non-explicit conditions that prevent the supporting action from having its intended effects. This new conception of mandatorily supporting preconditions through argument steps gives rise to a new and unique notion of threat. Under this new perspective, the concept of argument-argument threat in (23, 26) is now replaced by a broader notion of argument-argument threat that covers all the interferences that arise between the elements of a plan in which the qualification problem is addressed through the use of argument steps. Depending on where these argument-argument threats occur in the plan, we will distinguish between **threats** (Section 2.6.6.1) and **attacks** (Section 2.6.6.2).

2.5.3 Multi-Agent Planning Protocol

First, we outline the procedure followed by the DeLP-MAPOP protocol that interleaves a planning stage, an argumentation stage and a selection stage. Given a set of global goals, G, that address the requirements of an AmI application, agents build their own planning task $\mathbb{M}_x$ so they can differently contribute to the construction of the joint solution plan. The starting point of the MAP protocol is an empty initial plan Π_0 and the output is the solution plan. Once checked the plan Π is not a solution, the first step is to select an open goal $\Phi \in G(\Pi)$ of the planning task for resolution (choose^1 step in Algorithm 2). Then it

[1]The open goal Φ is selected as the most costly open goal according to a reachability analysis of the variables.

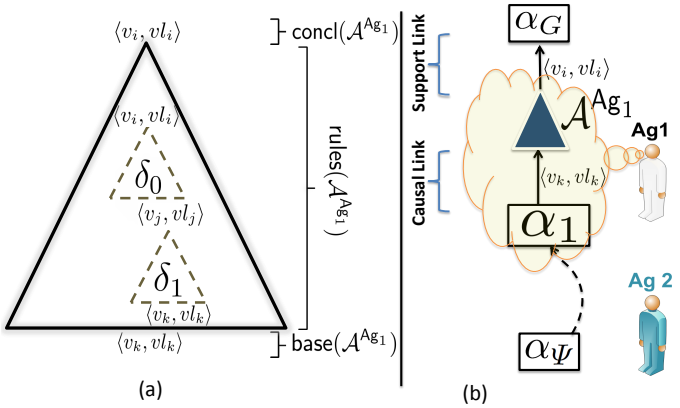

Figure 2.26: (a) An argument $\mathcal{A}^{Ag_1}$ for $\langle v_i, vl_i \rangle$ by using two defeasible rules: $\delta_0 = \{\langle v_i, vl_i \rangle\} -\!\prec \{\langle v_j, vl_j \rangle\}$ and $\delta_1 = \{\langle v_j, vl_j \rangle\} -\!\prec \{\langle v_k, vl_k \rangle\}$, such that $v_i \neq v_j$ and $v_j \neq v_k$ and $\{v_i, v_j, v_k\} \subseteq V_x$; (b) An example of a partial plan.

comes the planning stage (PROPOSALS step in Algorithm 2) where agents put forward and exchange different partial order plans that would potentially solve Φ. Following, agents get involved in an argumentative dialogue (EVALUATION step in Algorithm 2) in which they expose their arguments for or against each of the proposals. This evaluation process performs a *warranty procedure* to determine which proposals do not receive attacks or, otherwise, the received attacks do not succeed. Subsequently, ambient agents reach an agreement as to which about the next partial plan and they continue the search exploration (SELECTION step in Algorithm 2). The process is repeated until a solution plan is found.

The state-variable representation used in DeLP-MAPOP is based on the latest PDDL (Planning Domain Definition Language) version, PDDL3.1 (78), which was introduced in the context of the 2008 International Planning Competition. Here, we extend the language PDDL3.1 for supporting the specification of defeasible rules and the ambient agent's preferences. Moreover, our language allow us to specify binary variables. A state variable v_i is interpreted in PDDL3.1 as a function that represents a characteristic shared by some of the objects that define the problem. v_i is a tuple that takes the following form $v_i = (vN_i\ p_1 \ldots p_n)$, where vN_i is the unique variable's name and $p_1 \ldots p_n$ are the objects as input parameters of the function. For instance, let *pos-t11* be a variable that in-

input : The initial plan $\Pi_0 := \{\alpha_\Psi \prec \alpha_G\}$.
output: The solution plan Π.

$\Pi := \Pi_0$
while $\Pi <> null$ **do**
  if $G(\Pi) = \emptyset$ **then**
    return Π *[It is a plan solution.]*
  end
  else
    choose $\Phi \in G(\Pi)$;
    $\mathsf{Ref}(\Pi, \Phi) := \mathsf{PROPOSALS}(\Pi, \Phi)$;
    [Each plan Π_r of the set $\mathsf{Ref}(\Pi, \Phi)$ is a choice (partial-order plan) extending Π.]
    if $\mathsf{Ref}(\Pi, \Phi) = \emptyset$ **then**
      [Backtracking process.]
    end
    else
      $\mathsf{EVALUATION}(\mathsf{Ref}(\Pi, \Phi))$;
      $\Pi := \mathsf{SELECTION}()$;
    end
  end
end
return fail; *[Not exists plan.]*

Algorithm 2: Multi-agent planning protocol overview.

dicates the current position of the medical team *t11*; this variable is encoded in PDDL3.1 through the function *(pos t11)*, where '*pos*' is the function name and *t11* is the function parameter. An assignment of a value vl_i to a variable v_i in PDDL3.1 is denoted by $(\texttt{assign}\ \texttt{v}_\texttt{i}\ \texttt{vl}_\texttt{i})$; and the comparison operation is represented by $(=\ \texttt{v}_\texttt{i}\ \texttt{vl}_\texttt{i})$. We also allow to express multi-valued variables for ease of coding, denoted by $(\texttt{member}\ \texttt{v}_\texttt{i}\ \texttt{vl}_\texttt{i})$. For simplicity, we will use the notation <variable,value> in the explanations and use the PDDL3.1 language only to show the encoding of the defeasible rules and planning actions of the planning task. All of these encodings will be shown in a framed box labeled with the caption name Listing.

2.5.3.1 Overview of the Application Scenario

This section provides a brief overview of the AmI application upon which the framework DeLP-MAPOP is applied. The purpose is to motivate the interest of this type of applications as well as the utilization of a defeasible planning model to carry out the necessary operations to fulfill the user's need at a specific time.

Nowadays, more and more patients are suffering heart diseases which is the main cause of premature death. The monitoring of people suffering heart failure is currently a challenge for AmI systems. The work in (77) presents a first approach to use an AmI system with centralized planning capabilities for assisting patients suffering diabetics problems. Here, we assume that the patient's home is equipped with appropriate technologies to create the AmI environment. The patient is monitored with a system, in the form of a bracelet, which collects the patient's physical activity and wirelessly transmits it to a device responsible for monitoring patient's heart rate. When a need is detected by this device, e.g. an extremely lower level of a patient's physical activity which may end up in a heart attack, the AmI environment executes DeLP-MAPOP for assisting the patient until the health services arrive to the patient's home.

In this application, we have the following ambient agents: a communication agent in charge of using telecommunication devices such as a cell telephone to call the emergency services; the assistant agent, who is responsible for controlling an automated external defibrillator, an activity tracking device, a position tracking device, etc. to interact with both the environment and the user; and the transport agent, whose main function is to guide the ambulance/helicopter to follow the best path to reach the patient's home. Agents have different capabilities according to their role so they contribute to the overall plan with

different actions accordingly. However, we assume that agents' beliefs concern any aspect of the context information and so agents can make assumptions on the current status of the application regarding any type of information. That is, beliefs are not necessarily related to the planning capabilities of the agent, they can refer to any aspect of the AmI environment. The hospitals' preferences are associated with the transport-agent specific preferences, while the patient's preferences are related to the specific preferences of the assistant agent. For space reasons, we omit the specification of the planning task of each ambient agents.

2.5.3.2 Plan proposals process

At the PROPOSALS stage, agents generate their refinements $\mathsf{Ref}(\Pi, \Phi)$ to solve an open goal Φ in a partial plan Π, similarly to a plan-space planning process that builds a POP tree, except that each refinement or successor of Π may be now generated by a different agent. Another distinguishing characteristic of the partial order plans generated in DeLP-MAPOP is that they also contain argument steps, as explained in section 2.6.4.5, to support action preconditions; this argument structure formed in each partial plan will be later used in the EVALUATION process. The PROPOSALS stage finishes when all agents have made their plan proposals at their turn and these are communicated to the rest of agents. Then, agents update their set of actions with the information appearing in the refinements proposed by the other agents.

Let's suppose an ambient agent Ag_1 who has transport capabilities and knows there are three hospitals in the city $\{$*H1, H2, H3*$\}$. Each hospital disposes of two ambulances from *{a11, a12 . . ., a32}* (one equipped with an Advanced Life Support (ALS) equipment, and the other equipped with a Basic Life Support (BLS) equipment) and one emergency helicopter from *{h1, . . . h3}*. Moreover, Ag_1 knows there are always two emergency medical teams from the set *{t11, t12 . . ., t32}* on call in each hospital: one handles the ALS emergency equipment, and is formed by an ambulance driver, a nurse and a physician; the other handles the BLS equipment and is formed by an ambulance driver and a nursing assistant. Ag_1 also has the defeasible rule specified in Listing 2.10 and the planning action shown in Listing 2.9, among others. Note that the new location of the ambulance and the medical-team are generated through the defeasible rule moved-medical-assistance, which is embedded in an argument whose base must be supported by the effects of the action moving-medical-assistance. This allows agents to intervene during the argumentative dia-

logue in the EVALUATION stage to defeasibly attack the intended effects of the planning action; that is, in case agents have beliefs that make them conclude that the action would not achieve its expected effects.

```
(:def-rule moved-medical-assistance
    :parameters(?a - ambulance
    ?a1 address-hospital ?a2 - address-patient-home
    ?m - medical-team)
    :head (and (assign (at ?a) ?a2)
               (assign (pos ?m) ?a2))
    :body (and (= (moved-amb ?a ?a1) ?a2)
               (= (moved-team ?m ?a1) ?a2)))
```

Listing 2.1: The body of the defeasible rule matches the effects of the action moving-medical-assistance to deal with the qualification problem.

```
(:action moving-medical-assistance
   :parameters (?a - ambulance
    ?a1 address-hospital ?a2 - address-patient-home
    ?m - medical-team ?t - support-type)
    :effect (and (assign (moved-amb ?a ?a1) ?a2)
                 (assign (moved-team ?m ?a1) ?a2))
    :precondition (and (member (link ?a1) ?a2)
                       (member (type ?t) ?m)
                       (member (contains ?t) ?a)
                       (= (at ?a) ?a1)
                       (= (pos ?m) ?a1)))
```

Listing 2.2: An action for moving an ambulance from a location to other one.

Let pH be the patient's home. If Ag_1 is asked to solve the open goal $\mathsf{P}(\alpha_G) = \langle$ *at-?a, pH* $\rangle$ ('?a' is an ambulance) generated by the AMS agent to assist the patient, Ag_1 generates at least 6 refinement plans (3 hospitals * 2 ambulances) by using Listings 2.10 and 2.9 at the PROPOSALS stage. One of these proposed refinement plan generated is $\Pi_r^{\mathsf{Ag}_1}$, such that $\mathcal{OC}(\Pi_r^{\mathsf{Ag}_1}) = \{\alpha_\Psi \prec \alpha_1; \alpha_1 \prec \mathcal{A}^{\mathsf{Ag}_1}; \mathcal{A}^{\mathsf{Ag}_1} \prec \alpha_G\}$, as shown graphically in Figure 2.43(b); in this particular example:

- $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \{\langle$ *pos-t11, pH* $\rangle, \langle$ *at-a11, pH* $\rangle\}$ matches $\mathsf{P}(\alpha_G)$.
- $\mathsf{X}(\alpha_1) = \{\langle$ *moved-amb-a11-H1, pH* $\rangle, \langle$ *moved-team-t11-H1, pH* $\rangle\}$ matches $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$.

Therefore, argument $\mathcal{A}^{\mathsf{Ag}_1}$ is indirectly deriving the effects of the action α_1. However, unlike non-argumentative MAP systems, in DeLP-MAPOP the open goal $\langle$ *at-?a, pH* $\rangle$ can also be derived by means of an argument that an agent, say Ag_2, puts forward to indicate, that according to its knowledge, ambulance *a31* is already at the patient's home. The base of this argument may be supported with the information provided by an ambulance position tracking device which allows Ag_2 to infer that an ambulance is already located at the patient's home. As the rest of agents do not own this information, they would claim for the inclusion of an action that moves an ambulance to the indicated place. Therefore, unlike classical planning, the argumentation mechanism in DeLP-MAPOP enables supporting an open goal with the context information of an agent without having to necessarily include an action to satisfy such goal, which results in less costly plans. The next stage would show the procedure to guarantee that a goal is satisfactorily warranted by an argument.

2.5.3.3 Plan evaluation process

At the EVALUATION stage, agents become engaged in a number of argumentative dialogues aimed at evaluating the guarantee of a successful execution of a plan proposal, i.e the possibility that the actions' intended effects or the derived information, both represented as argument steps in the plan proposal under evaluation, are not achieved as a result of AmI environment changes.

The input of this process is $\mathsf{Ref}(\Pi, \Phi)$, the set of plans proposed by the agents at the previous plan proposal stage. Since ambient agents may have different available context information (represented as a combination of facts and defeasible rules) depending on their information sources, they may not agree on the evaluation of a plan proposal at some point during the dialogue. The EVALUATION stage generates as many argumentative dialogues as argument steps are present in the proposal plan under evaluation. An argumentative dialogue is an exchange of arguments for or against the fulfillment of an argument step, represented as a Plan Argument Dialogue (PAD) tree $\mathcal{T}^{\mathcal{A}}_{\Pi_r}$, where $\Pi_r \in \mathsf{Ref}(\Pi, \Phi)$ is the refinement plan to be evaluated and $\mathcal{A} \in \mathcal{AR}(\Pi_r)$ is the particular argument step to be evaluated. We denote the nodes in a PAD tree as tuples of the form (Π_r, $\mathcal{A}$, Γ), where Γ is a set of attacking arguments (whose bases are warranted in the plan Π_r) that will finally determine if argument $\mathcal{A}$ is warranted in plan Π_r. Every node in a PAD tree (except the root) represents a defeater of its parent, and the leaves of the tree

correspond to undefeated plans. The set of direct successors nodes of a given node Π_r, is denoted as $\mathsf{succ}(\Pi_r)$. More specifically:

1. The root of the tree is labeled with $(\Pi_r, \mathcal{A}, \emptyset)$.
2. A plan node $(\Pi_r, \mathcal{A}, \{\mathcal{B}\}) \in \mathsf{succ}(\Pi_r)$ represents an attack against the argument $\mathcal{A}$ in plan Π_r through the inclusion of an attacking argument, namely $\mathcal{B}$. Consequently, each node in $\mathsf{succ}(\Pi_r)$ stands for a defeater of the root argument $\mathcal{A}$, i.e. $\mathcal{B}$ is a defeater of $\mathcal{A}$.
3. A plan node $(\Pi_r, \mathcal{A}, \{\mathcal{B}, \mathcal{C}\}) \in \mathsf{succ}(\mathsf{succ}(\Pi_r))$ indicates an attack to the argument child $\mathcal{B}$ of the parent node through the inclusion of a new attacking argument, say $\mathcal{C}$, so this new node is a supporter of the root argument $\mathcal{A}$.

Informally we might see a PAD tree for an argument step $\mathcal{A}$ as generating a dialectical tree (24) for $\mathcal{A}$. But in DeLP-MAPOP the nodes in the PAD tree are contextualized within a plan. Every linear path from the root to a leaf corresponds to one different acceptable argumentation line. Circular argumentation (also known as fallacious argumentation) is avoided by applying both conditions from (24): no argument can be reintroduced in the same argumentation line and argument concordance must be guaranteed.

Let Ag_2 be an agent that has the defeasible rules detailed in Listing 2.14, and $\{\langle$*device-measure-the-traffic-H1-pH, high* $\rangle, \langle$ *maps-google-distance-H1-pH, long* $\rangle\} \subseteq \Psi_{\mathsf{Ag}_2}$. When Ag_1 sends the PAD tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ (containing only the root node) to the rest of agents, Ag_2 puts forward an attacking argument $\mathcal{B}^{\mathsf{Ag}_2} = (\{\langle$ *pos-t11, H1* $\rangle\}, \{\delta_0; \delta_1; \delta_2\})$, inspired by Listing 2.14, where:

- δ_0 =*(and* $\langle$ *pos-t11, H1* $\rangle$ $\langle$ *at-a11, H1* $\rangle$*)* $\prec$ *(and* $\langle$ *moved-amb-a11-H1, pH* $\rangle$ $\langle$ *moved-team-t11-H1, pH* $\rangle$ $\langle$ *traffic-jam-between-H1-pH, true* $\rangle$ $\langle$ *is-far-from-H1-pH, true* $\rangle$*).*
- δ_1 =$\langle$ *traffic-jam-between-H1-pH, true* $\rangle$ $\prec\langle$ *device-measure-the-traffic-H1-pH, high* $\rangle$.
- δ_2 =$\langle$ *is-far-from-H1-pH, true* $\rangle$ $\prec$ $\langle$ *maps-google-distance-H1-pH, long* $\rangle$.

which attacks $\mathcal{A}^{\mathsf{Ag}_1}$. Unlike agent Ag_1, agent Ag_2 knows that traffic jam is expected according to a smart device from the AmI system that monitors the traffic density be-

tween the hospital *H1* and the patient's home *pH*, and also knows that the distance between them provided by a web mapping service as Google Maps, is rather large. Both informations may be a reason to believe that an ambulance, initially located at the hospital *H1* will not arrive to *pH* in time for assisting the patient. Thus, Ag_2 creates a new node $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}\}) \in \mathsf{succ}(\Pi_r^{\mathsf{Ag}_1})$ among others, and sends it to rest of agents.

```
(:def-rule moved-medical-assistance-denied
    :parameters(?a - ambulance
    ?a1 address-hospital ?a2 - address-patient-home
    ?m - medical-team)
    :head (and (assign (at ?a) ?a1)
               (assign (pos ?m) ?a1))
    :body (and (= (moved-amb ?a ?a1) ?a2)
               (= (moved-team ?m ?a1) ?a2)
               (= (traffic-jam-between ?a1 ?a2) true)
               (= (is-far-from ?a1 ?a2) true)))
(:def-rule traffic-jam
    :parameters(?a1 address-hospital
    ?a2 - address-patient-home)
    :head (assign (traffic-jam-between ?a1 ?a2) true)
    :body (= (device-measure-the-traffic ?a1 ?a2) high))
(:def-rule distance
    :parameters(?a1 address-hospital
    ?a2 - address-patient-home)
    :head (assign (is-far-from ?a1 ?a2) true)
    :body (= (maps-google-distance ?a1 ?a2) long))
```

Listing 2.3: Defeasible rules for representing situations in which the ambulance may not arrive on time.

In the next round of the dialogue, $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}\})$ is received by the ambient agent Ag_3 who discovers a new attacking argument $\mathcal{C}^{\mathsf{Ag}_3}$ that defeats $\mathcal{B}^{\mathsf{Ag}_2}$, which is based on Listing 2.15.

```
(:def-rule carpool-lane
    :parameters(?a1 address-hospital
    ?a2 - address-patient-home)
    :head (assign (traffic-jam-between ?a1 ?a2) false)
    :body (= (carpool-lane-between ?a1 ?a2) true))
```

Listing 2.4: The defeasible rule used for representing a carpool lane which may prevent an ambulance from being stuck by a traffic congestion situation.

Assuming that $\langle$ *carpool-lane-between-H1-pH, true* $\rangle \in \Psi_{\mathsf{Ag}_3}$, then $\mathcal{C}^{\mathsf{Ag}_3} = (\{\langle$ *traffic-jam-between-H1-pH, false* $\rangle\}, \{\langle$ *traffic-jam-between-H1-pH, false* $\rangle \prec \langle$ *carpool-lane-*

between-H1-pH, true $\rangle\}$). That is, Ag_3 knows that there is a carpool lane (as an express lane) between *H1* and *pH*, which is a reason to believe that the ambulance *a11* can skip the traffic congestion on the way to reach the patient's home. Ag_3 creates a new plan $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}, \mathcal{C}^{\mathsf{Ag}_3}\})$ extending $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}\})$ with $\mathcal{C}^{\mathsf{Ag}_3}$, and sends it to the rest of agents. The evaluation dialogue for $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ continues until all defeaters are put forward in a round.

In order to check whether the argument of the root node is defeated or undefeated, the following procedure on the PAD tree is applied: label with a *U* (for *undefeated*) each terminal plan in the tree (i.e. each plan with no defeaters at all). Then, in a bottom-up fashion, we label a node with: *U* if each of its successors is labeled with a *D*; and *D* (for defeated) otherwise.

A plan in $\mathsf{Ref}(\Pi, \Phi)$ is labeled as an **undefeated refinement plan** if all the root plans of its PAD trees are labeled as undefeated. Otherwise the plan is provisionally labeled as a **defeated refinement plan** in the POP tree. Undefeated plans are obviously preferred over defeated plans as they represent a plan with no expectation failures according to the ambient agents. Nevertheless, defeated plans are maintained in the POP tree as their arguments may become later undefeated as the problem evolves and information changes. Finally, each ambient agent updates its initial facts and defeasible rules with the facts and defeasible rules from the exchanged arguments' bases.

2.5.3.4 Plan selection process

At the SELECTION stage, the aim is to select the next plan Π to be refined and continue with the plan-space planning process of the PROPOSALS stage, unless Π is already a solution in which case the DeLP-MAPOP protocol stops.

For selecting a plan, agents apply three criteria in order of priority over the set of evaluated plans from the previous stage. The objective is to select a plan considering a compromise between the desire to minimize the computational overhead and that of maximizing the quality of the solution plan. The three criteria are: first, the system applies a warranty procedure to discard the plans evaluated as defeated in the evaluation stage. Second, a heuristic function is applied over the undefeated plans resulting from the above filtering. We use two of the most popular heuristics in planning: SUM and MAX heuristics (79). The SUM heuristic estimates the cost of a plan as the sum of the cost of the pending open goals in the plan whereas the MAX heuristic returns the value of the

most costly open goal as heuristic estimation. Plans whose heuristic estimation is below a certain threshold are discarded from consideration. Finally, the last filtering over the remaining plans considers the preference functions. We have implemented two intersection techniques aimed at selecting the most preferable plan by the ambient agents according to their preferences. The first mechanism selects the plan whose actions are all among the preferences of every agent with a degree of preference above a certain threshold. If the application of this method returns an empty list then we compute the number of preferred actions in each plan and we select the plan with the largest proportion of preferred actions by the ambient agents.

2.5.4 Experimental Evaluation

The purpose of this section is to test the overall performance, scalability and quality of DeLP-MAPOP versus a MAP system with no argumentation (MAPOP) which has also been implemented in the same agent platform, and discuss the benefits and limitations of each system. We carried out several experiments considering three different levels of difficulty of the planning problem: small (composed by 8 grounded actions and 50 grounded defeasible rules), medium (composed by 16 grounded actions and 100 grounded defeasible rules) and large (composed by 24 grounded actions and 150 grounded defeasible rules). We used teams of agents of different size ranging from 1 (single-agent) to 5. We performed several tests varying the number of agents of each type in the AmI environment, namely transportation, communication and assistant agents, and we took the median values over 20 repetitions for each set of experiments with 'n' agents, regardless the type of agent. We used the MAX heuristic and the Intersection function.

DeLP-MAPOP and MAPOP are implemented on Magentix2[1], a multi-agent platform based on Apache Qpid[2], an open-source implementation of Advanced Message Queuing Protocol for communication.

With regard to scalability and performance, Figures 2.27(a), 2.27(b) and 2.27(c) show the average time spent on each stage of the DeLP-MAPOP protocol, while Figure 2.27(d) shows the average total time to find a solution plan, including parsing the problem file and grounding the planning actions and defeasible rules. The horizontal axis (the same for the rest of the figures) depicts the size of the team of ambient agents, while the vertical axis

[1] http://www.gti-ia.upv.es/sma/tools/magentix2/index.php
[2] http://qpid.apache.org/

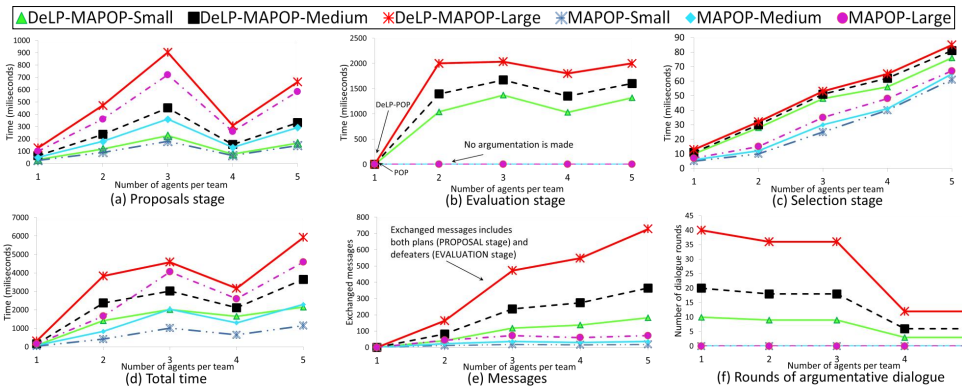

Figure 2.27: Performance measures.

displays the time in milliseconds. As expected, the average time spent in DeLP-MAPOP is always greater than the time spent in MAPOP due to the following reasons: i) in the PROPOSALS stage, the ambient agents from DeLP-MAPOP do not only have to reason about which actions would achieve the selected open goal, but also need to reason about which arguments would support it; ii) the EVALUATION stage is not considered in MAPOP; and iii) the SELECTION stage is replaced in MAPOP by a single heuristic function. It is also noticeable that the more agents in a team, the more exchanged messages between them, causing each stage to take longer in DeLP-MAPOP. Figure 2.27(e) illustrates precisely that, as the number of agents increases, the number of exchanged messages is larger; Figure 2.27(f) shows that as the size of the team increases, the number of dialogue rounds is lower because in this case more attacking arguments tend to appear in a single round, thus decreasing the number of rounds.

Figure 2.28 shows the evaluation of the quality of the obtained solution plans. Figure 2.28(g) shows that the average number of action steps in solution plans of DeLP-MAPOP is lower or equal than the average number in solution plans of MAPOP. The reason is that in MAPOP, an open goal that is not a threat can only be achieved through an action step, while in DeLP-MAPOP the open goal can also be supported by an argument step whose base is already guaranteed in the plan. In these cases, the cost of the DeLP-MAPOP plans is smaller because fewer actions are required to support the open goals, meaning that the agents' beliefs support the fulfillment of a goal without explicitly including an

additional action in the plan. The fact that argument steps are not used in MAPOP is precisely shown in Figure 2.28(h). On the other hand, we can see in Figure 2.28(i) a comparison of the quality of plans generated with a single-agent team versus plans generated by teams with more than one agent. Obviously, in the first case, plans are sequential while DeLP-MAPOP returns plans with parallel actions that can be simultaneously executed by different ambient agents.

We also carried out one more experiment: which action steps in MAPOP solution plans are actually discarded during an argumentative dialogue in DeLP-MAPOP plans. This latter aspect is also a very relevant issue as we wanted to compare the plans returned by both systems and see how many plans, and actions correspondingly, of MAPOP were actually discarded by the agents in DeLP-MAPOP during the argumentative dialogues. The results of this experiment are shown in Figure 2.28(j). As can be seen, according to the knowledge of the ambient agents, 0% of the solution plans generated by DeLP-MAPOP comprise failing actions, i.e. actions whose intended effects were acknowledged to fail at the EVALUATION stage. Obviously, as long as agents acquire more information from the context, argumentative dialogues will fit reality better and, therefore, the guarantee of a successful solution plan (a plan with no expected failures) would also be greater. Furthermore, this experiment allowed us to check the correctness of the argumentative dialogues at the EVALUATION stage. However, in the case of MAPOP, up to 50% of the plans had actions that were discarded by the ambient agents in DeLP-MAPOP, that is, actions that agents acknowledged that would not be successfully executed.

Examining the influence of preferences in DeLP-MAPOP, Figure 2.28(k) shows that the average satisfaction of each team with the solution plans decreases as the size of the team increases. We calculated the satisfaction of an individual agent on a solution plan by averaging its preferences in the action steps of the plan, while the team satisfaction is calculated as the average of the individual satisfactions. Figure 2.28(l) shows that the difference of satisfaction between agents tends to increase as the size of the teams also increases. It is desirable that the difference is as small as possible for that all agents are equally satisfied.

2.5.5 Conclusions and Future Work

This paper presents the specification, implementation and an exhaustive experimentation of DeLP-MAPOP, an argumentation-based defeasible planning framework, on AmI ap-

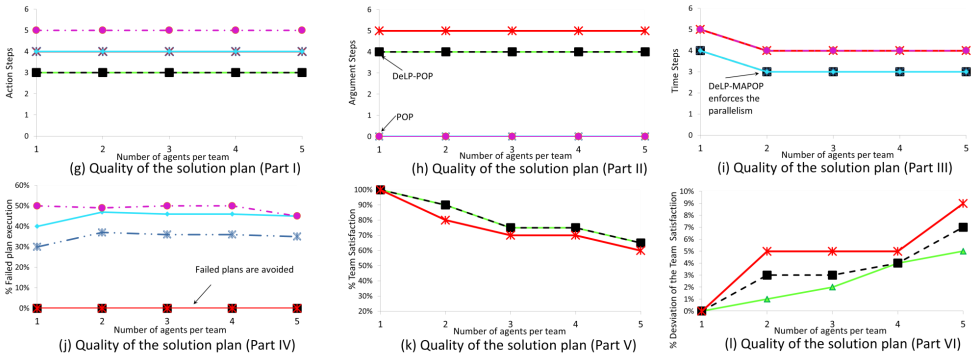

Figure 2.28: Quality measures.

plications. Our most relevant contribution is a fully implemented MAP framework that has been extensively tested in AmI environments. DeLP-MAPOP realizes three independent but cooperative processes to propose, criticize, defend and select alternative plan proposals. The results show two advantages of DeLP-MAPOP over a MAP process with no argumentation: (i) since each plan step of a plan proposal is collaboratively argued, DeLP-MAPOP returns plans whose actions are not likely to fail at execution time according to the information and beliefs of the ambient agents; and (ii) since open goals can also be supported by argument steps whose base is warranted with the facts of the plan, the context information and defeasible reasoning of agents provide a means to satisfy goals of the problem without an explicit inclusion of a planning action; this avoids considering unnecessary action steps thus reducing the total cost of the plan.

As future work, we intend to test the effectiveness and feasibility of DeLP-MAPOP in a hospital pilot program, as well as an extension to temporal defeasible argumentation for MAP (65). Currently, we are working on the development of a more elaborated heuristic function that (i) analyzes the transitions between the values a state variable can take, and (ii) considers the experiences from the plan evaluation process (case-based argumentation) to predict the potential number of attacks that a plan can receive. We are also interested in studying the influence of the trust on the sources (devices) used by the ambient agents to acquire the context information as well as how a trust level determines the conflict resolution between attacking arguments. Finally, a comparison with other MAP approaches will be considered.

Acknowledgements

This work is mainly supported by the Spanish Ministry of Science and Education under the FPU Grant reference AP2009-1896 awarded to Sergio Pajares Ferrando, and projects, TIN2011-27652-C03-01, Consolider Ingenio 2010 CSD2007-00022, and PROMETEO/2008/051.

2.6 Selected Paper 5: Context-Aware Multi-Agent Planning in Intelligent Environments (INS Journal 2013)

Abstract. *A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use. Multi-agent planning generalizes the problem of planning in domains where several agents plan and act together, and share resources, activities, and goals. This contribution presents a practical extension of a formal theoretical model for Context-Aware Multi-Agent Planning based upon an argumentation-based defeasible logic. Our framework, named* CAMAP*, is implemented on a platform for open multi-agent systems and has been experimentally tested, among others, in applications of ambient intelligence in the field of health-care.* CAMAP *is based on a multi-agent partial-order planning paradigm in which agents have diverse abilities, use an argumentation-based defeasible contextual reasoning to support their own beliefs and refute the beliefs of the others according to their context knowledge during the plan search process.* CAMAP *shows to be an adequate approach to tackle ambient intelligence problems as it gathers together in a single framework the ability of planning while it allows agents to put forward arguments that support or argue upon the accuracy, unambiguity and reliability of the context-aware information.*

2.6.1 Introduction

Context-aware is concerned with the acquisition of context information, the abstraction and understanding of this information, and application of behaviour based on the recognized context information(80, 81). Much of the work on context-aware has been focused more deeply on perceiving the context as a matter of user location. This trend has put the emphasis on research fields as location awareness (82) and on the manifold ways, such as sensors, network information or smart devices, to extract the context information. However, in the last few years, the notion of context-aware has been extended to describe not only the ability of the computer to sense and extract information in the field, but also to enable selective responses such as triggering actions based on context (83).

The work we present here lays on the adaptive and intelligent behavior of context-aware systems; particularly, on the contextual behavior of Ambient Intelligence (AmI) applications. AmI is an emerging discipline that brings intelligence to our everyday environments and makes those environments sensitive to us. In AmI environments, technol-

ogy becomes invisible and people are surrounded with networks of embedded intelligent devices that can sense the available context information, anticipate, and adapt to their needs (74, 84). As an example, Easishop is an ubiquitous commerce application for assistance in everyday shopping that emerges as a juxtaposition of AmI and e-commerce (85). Easishop autonomously and proactively provides assistance to the user by seeking out shops that sell the items on the user's shopping list. To realize such ubiquitous applications with optimal usability, context-aware behaviour is seen as the key enabling factor.

The use of agent technology is becoming very popular in AmI applications because they can be seen as involving entities as autonomous agents that extract, process, change and share the available context information (76). While software agents have been used before for building AmI middleware, hardware agents are recently being used for the design of multi-agent architectures for AmI Systems (86) or in multi-agent based simulations for testing and validating large-scale AmI systems in dangerous environments (87). We can find various examples of AmI systems concerning the coordination of agents associated to devices in order to resolve complex tasks that no agent can do by itself; for instance, the AmbieAgents infrastructure (88) for context-based information services, or the SpacialAgents platform (89) which uses mobile agents to offer services on the user's devices when the user enters a place that offers certain capabilities. In general, in all these applications, ambient agents are entities associated to a device which extract raw context-data, offer services and share the context information, but they are not endowed with reasoning capabilities to achieve the desired context-aware behavior. Thus, agents are simply conceived as sources of information that extract, share and exchange data but contextual adaptive behaviour is usually designed as a single-agent and independent process that takes as input the context-data collected by the ambient agents. This is reasonable if we consider that in most applications ambient agents are not other than device agents.

We want to exploit the use of agent technology in AmI applications, particularly on the field of health-care problems (90). Homecare applications are becoming very popular, above all among elderly people (91), because they allow synchronizing an electronic agenda with a central system that collects the daily reports and interfaces the central systems. However, we want to go a step further and design ambient agents not only as device agents associated to blood pressure or temperature monitors, but as entities like doctors or nurses that receive the monitor readings, perform context inferences to find out the

real patient's conditions and exchange their findings for a more accurate diagnosis. Put another way, the final objective is to have agents involved in designing a plan to stabilize a patient accordingly to their raw context-data and context inferences. Thus, we explore the application of Artificial Intelligence planning techniques (36) as planning is a desired ability in AmI systems in order to achieve a goal-oriented behavior. More particularly, our aim is to apply multi-agent planning (92) to decide the course of action to meet the needs of the patient.

Very few applications are actually able to offer a plan for fixing anomalies detected during monitoring the vital signs of a patient. Amigoni et al. present a centralized planner to assist a diabetic patient (77) where device agents do not take part in the planning activity but they are simply device agents. Therefore, agents in (77) are not able to do context inference nor distributed planning. It is also important to remark the imperfect nature of the context in AmI environments, the inherently distributed nature of health-care assistance, where several agents intervene in the problem, and the heterogeneity of local context theories. Thus, it is required to introduce mechanisms for agents to exchange, discuss and solve the potential conflicts that may arise from the interaction of different contexts (93). We opt for using *argumentation* (94) as a defeasible reasoning mechanism that will allow agents to defeasibly support their decisions, interact to each other and come up with a joint solution plan for the patient.

In this paper, we present Context-Aware Multi-Agent Planning (CAMAP), an approach for multi-agent planning that applies argumentation mechanisms to decide the most appropriate course of action according to the context information distributed among the agents. CAMAP is applied to a real-world application of AmI in the field of health-care. The remainder of this paper is divided as follows. After a brief review of the related work and background, we introduce the basic elements of the CAMAP system. We then describe a real-life AmI scenario to deal with a person suffering from a heart disease. Following, we present the CAMAP protocol applied to the AmI scenario. The next section presents the experiments carried out to test and validate the planning model. Finally, the last section concludes and presents some directions for future work.

2.6.2 Related Work

In this section, we briefly review related work on AmI in health-care, multi-agent planning and argumentation, the three main topics related to CAMAP. We only focus on works

that combine at least two of the above topics, without paying special attention to the large amount of specific works on each research topic.

Argumentation can be viewed as a powerful tool for reasoning about incomplete and inconsistent contextual information through a rational interaction of arguments for and against some conclusion derived by an ambient agent. An argument is a chain of reasoning that concludes one piece of information (conclusion) on the basis of some other pieces of information (premises). Argumentation has been successfully proved in AmI applications as exposed by A. Bikakis (76) and P. Moraitis (95). Specifically, A. Bikakis proposes an argumentation model to decide if a context-aware mobile phone should ring (in case of incoming calls) based on the context of the owner of the phone. In (95), authors propose a different argumentation model to decide the type of public transportation that a person, who uses a wheelchair and has heart problems, should take according to the specific context. The main idea of these relevant works is very similar; they use context information for building arguments for or against a given action in a given context. Their reasoning process is focused only on one action but not on a course of action (planning).

The work presented by D.R. García (23) combines argumentation (as a mechanism for reasoning about context) and a centralized planner that follows a Partial-Order Planning (POP) approach (43) although this framework has not been experimentally tested in real applications yet. Unlike (23), F. Amigoni presents a planner applied to an AmI environment in the field of health-care (77), which is used to monitoring and responding to the needs of a diabetic patient. More specifically, F. Amigoni assumes that only one agent is endowed with planning abilities, but considers several device agents which have no responsibilities in building the plan due to its limitations in processing and communication. The work in (77), however, does not use argumentation. On the other hand, the work of A. Bahrani (96, 97), which follows a mixed initiative planning approach (98), proposes a planning model that enables to analyze contextual information exclusively referred to military situations. Specifically, it uses a graphical tool to visualize the contextual information through which the user may decide whether this information should be taken into account in the planning for military tasks. All the aforementioned approaches present an important limitation; they are centralized planning models which do not allow automated reasoning with multiple agents in a distributed way.

The motivation for introducing distributed reasoning in a multi-agent environment is threefold. First, multi-agent systems can be beneficial in many domains, particularly when

a system is composed of multiple entities that are distributed functionally or spatially (99, 100). Second, a multi-agent system allows for realizing multiple device agents that read raw context data and perform their own context inferences on the basis of the monitor readings as well as improve the ability to more rapidly detect context information changes. Third, distributed execution promotes the efficiency of parallel processing of actions, the robustness of the system to cope with complex planning problems and the simplicity of an incremental construction across a network of interconnected agents, thus avoiding the critical failures and resource limitations of centralized systems. In Multi-Agent Planning (MAP), an ambient agent is usually executed on an independent host and can encompass several devices. This increases the communication capacity as well as autonomy and endow agents with the necessary abilities to pose a goal and build a plan for this goal. Typically, MAP approaches have been conceived as:

(1) *Plan Selection*: Agents construct independent plans for the same common goal and a centralized algorithm is used to select the best solution plan. In this case, MAP emphasizes the problem of selecting the best solution plan.

(2) *Plan Merging*: Agents construct independent plans for different subgoals and a centralized algorithm is used to merge these plans. In this case, MAP emphasizes the problem of controlling and coordinating a posteriori local plans of independent agents.

(3) *Plan Construction*: Agents propose iteratively refinements to a base plan until a consistent joint plan that solves the problem goals is obtained. In this case, MAP is about an incremental construction of a joint plan among several agents that have incomplete views of the world, which prevents them from coming up with complete local plans for themselves.

With respect to (1), A. Belesiotis (29) proposes dialogue protocols that enable agents to discuss candidate plans and reach agreements. Specifically, they introduce the defeasible situation calculus, a novel formalism based on the combination of situation calculus and defeasible logic programming for reasoning about plans based on contradictory planning beliefs. Secondly, with respect to (2), one of the most well-known approaches for the coordination of plans is the partial global planning framework (101) and its extension, the generalized partial global planning approach (102). In addition, A. Toniolo (31, 103)

presents an argumentation-based model for deliberative dialogues based on argumentation schemes. This work focuses on conflicts among the plans of agents, which may be caused by concurrent actions, plan constraints or norms the agents must adhere to. Finally, as we will see in Section 2.6.3, the works in (26, 34, 73) follow a MAP approach based on (3).

The benefits and advantages of (3) with respect to (1) and (2) are widely discussed in the works (33, 104). Basically, the MAP model (3) can be successfully applied in problems where agents have little iteration to each other to solve the planning problem (loosely-coupled problems) as well as problems where agents have necessarily to interact to engage their respective sub-plans (tightly-coupled problems). The underlying principle of this general-purpose MAP model is interleaving planning and coordination continuously. It is empirically demonstrated in (33) that (1) and (2) are not appropriate approaches for solving tightly-coupled planning problems. In order to be able to solve any type of planning problem, CAMAP relies on a MAP model (3). On the other hand, none of the works that follows one of these three MAP approaches have been designed and tested to cope with the requirements of AmI applications in health-care.

Table 2.2 summarizes the approaches presented in this section. As we can see, the novelty of this contribution is the combination of all the topics into the same model, which results in a CAMAP system for intelligent environments. To the best of our knowledge there are no works that combine all of these topics, let alone implementations or empirical evaluation.

	Ambient Intelligence	Health-Care System	Argumentation	Multi-Agent System	Planning
F. Amigoni	Yes	Yes	No	No	Yes
A. Bikakis	Yes	No	Yes	Yes	No
A. Bahrani	Yes	No	No	No	Yes
A. Belesiotis	No	No	Yes	Yes	MAP (1)
A. Toniolo	No	No	Yes	Yes	MAP (2)
D.R. García	No	No	Yes	No	Yes
P. Moraitis	Yes	No	Yes	Yes	No
Our proposal	Yes	Yes	Yes	Yes	MAP (3)

Table 2.2: Summary of the related work.

2.6.2.1 Contributions of our model

This paper contributes with the design, implementation and evaluation of a model for **Context-Aware Multi-Agent Planning** (CAMAP) particularly applied to health-care scenarios in AmI environments, which uses a special type of argumentation, known as defeasible argumentation (75). Particularly:

a) CAMAP tackles planning problems in AmI environments where several ambient agents extract raw context-data, make context inferences, and cooperate in the planning process.

b) The model can be used in many real applications where the capabilities of perceiving the context and planning are distributed across the ambient agents.

c) CAMAP is capable of solving problems with different levels of complexity and coupling level (loosely-coupled and tightly-coupled).

d) Agents involved in the problem put forward their opinions about the plan construction by using defeasible argumentation. This allows agents to gradually interact and discuss the impact and consequences of the context information in the actions of the plan and present arguments for or against a particular action choice.

e) Agents are able to take the best action choice for the plan under construction by taking into account the context information of the other agents.

2.6.3 Background

In this section, we summarize the foundations and previous works which the present contribution is based on. Our framework, CAMAP, builds upon the formalism DeLP (24) for the definition and specification of the defeasible argumentation mechanism. DeLP, a defeasible logic programming formalism, is one of the most popular approaches to make context inferences by using defeasible argumentation. The key element of DeLP are the *defeasible rules* (Head $\prec$ Body), which are used to represent a deductive relation between pieces of knowledge that could be defeated once another piece of knowledge is considered. For instance, a defeasible rule like *emergency* $\prec$ *patient-high-fever* represents the belief of an ambient agent that if he holds a fact that the monitor reading returns the patient has high fever then there are provable reasons to declare an emergency. The defeasible

rule $\sim$*emergency* $\prec$ *{normal-pulse, conscious, correct-breathing}* provides reasons to believe the contrary, in whose case we say that the first piece of information is acknowledged to fail in case *{normal-pulse, conscious, correct-breathing}* hold in the context. However, assuming that another ambient agent knows that the patient is vomiting blood, i.e., {*bloody-vomit*} holds in the context, then he might derive the patient does not have a normal pulse by following the defeasible rules {$\sim$*normal-pulse* $\prec$ *internal-bleeding*; *internal-bleeding* $\prec$ *bloody-vomit*}, which entails an attack to the defeasible rule whose conclusion is $\sim$*emergency*. Thus, arguments are combinations of defeasible rules and facts which may result in conflicting pieces of information. In this case, arguments are evaluated and compared to each other to decide which one prevails.

Partial-Order Planning (POP) is a suitable planning approach to address the requirements derived from a distributed planning thanks to the application of the least commitment principle (43), which delays commitment of action orderings until such commitments become necessary to resolve inconsistencies. In POP, a plan is represented as a set of actions and a set of ordering constraints defining a partial order between actions. The POP paradigm has been widely used in architectures for planning and execution, information gathering, planning and scheduling or temporal planning. In (54), authors argue that POP-based frameworks offer a more promising approach for handling domains with durative actions and temporal and resource constraints as compared to other planning approaches. Even for simple planning tasks, partial-order planners offer a higher degree of execution flexibility. For these reasons, the underlying planning model of ambient agents in CAMAP is a POP planner.

A theoretical extension of POP with DeLP-style argumentation, denoted as DeLP-POP, was introduced in (23), where both actions and arguments may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. Unlike actions, arguments are not only introduced to intentionally support some step of a plan, but they are also used to defeat or defend other supporting arguments in the plan. When actions and arguments are combined in a partial-order plan, new types of interferences or threats appear, which need to be identified and resolved in order to obtain valid plans (23).

The works in (26, 34, 73) propose a theoretical extension of the DeLP-POP to a multi-agent environment. Specifically, these works present a dialogue based on a logic formalism for argumentative plan search, by which agents exchange plan proposals and

arguments for or against such proposals, taking into account their context information. To the best of our knowledge, these theoretical works have neither been implemented nor tested on real-world domains like AmI applications.

This contribution presents CAMAP, a context-aware system that combines, implements and evaluates features of multi-agent defeasible argumentation and multi-agent planning in AmI applications. CAMAP extends the theoretical works in (26, 34, 73) with some additional capabilities to account for the requirements of a context-aware system. Finally, CAMAP is implemented and experimentally tested in a real-world context as a health-care application.

2.6.4 Definition of Components of the Context-Aware System

In this section, we provide definitions for all the elements used in the CAMAP framework.

2.6.4.1 Ambient Agents and Ambient Artifacts

In CAMAP, ambient agents act as planning agents that hold different beliefs encoded in the form of defeasible rules and facts, and capabilities encoded as planning actions. Thus, we assume that the capabilities to extract the raw context-data, make context inferences, and plan actions are distributed across the ambient agents.

An ambient artifact is an entity which acts as a mediator between an ambient agent and a smart device of the environment (105). The aim of artifacts is to allow agents to easily interact with the context in order to: (a) extract the context information and encode it in the form of facts that will be then provided to the agents; (b) achieve a goal-oriented behavior, by which artifacts generate goals as input to the planning process; and, (c) execute actions of a solution plan. We distinguish between *informative artifacts* for the tasks (a) and (b) and *executor artifacts* for the task (c). In CAMAP, an ambient agent interacts with one or more artifacts.

2.6.4.2 Context-Aware Information

The representational scheme used by CAMAP to model components of the AmI environment is based on a state-variable representation, where variables map to a finite domain of values which represent the problem objects. A state-variable representation is equivalent to a classical planning representation in expressive power and it is also useful in

non-classical planning problems as a way to handle numbers, functions and time. In this paper, we will restrict our attention to only non-numeric variables. Since planning actions change the state of the world and defeasible rules make assumptions about the state of the world, actions and defeasible rules are most naturally modeled as elements that change the values of the state variables. The variable-value pair $\langle v_i, vl_i \rangle$ denotes that the value vl_i is assigned to the variable v_i. For instance, the variable-value pair $\langle$ *at-amb, pH* $\rangle$ indicates that the location of the ambulance *amb* is the patient's home, *pH*; that is, the value of the variable denoting the position of the ambulance is the patient's home.

In what follows, we define the elements used to represent the agent's context information:

(i) The set of objects, O, represents the elements of the planning task involved in the planning actions and defeasible rules.

(ii) The set of state variables, V, are used to model the state of the world. Each state variable $v_i \in \mathsf{V}$ is mapped to a finite domain of mutually exclusive values D_{v_i}, where $\forall v_i \in \mathsf{V}$, $D_{v_i} \subseteq \mathsf{O}$.

(iii) The initial state of the problem, Ψ, is a consistent set of facts, represented as variable-value pairs. A variable which has not been assigned a value in the initial state is assumed to have an *unknown* value.

(iv) The set of defeasible rules, Δ, where each rule δ follows the form $\langle \mathsf{head}(\delta) \prec \mathsf{body}(\delta) \rangle$. If the set of variable-value pairs in $\mathsf{body}(\delta)$ is warranted, i.e., if the variables that model the problem state have the values specified in $\mathsf{body}(\delta)$, then δ is applicable in such state, and for each $\langle v_i, vl_i \rangle$ that appears in the head of the rule, v_i is assigned the value vl_i.

(v) The set of planning actions, A, such that an action is represented as $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$, where $\mathsf{P}(\alpha)$ is a set of preconditions encoded as variable-value pairs that must be satisfied in the problem state in order to achieve the effects in $\mathsf{X}(\alpha)$, also encoded as variable-value pairs $\langle v_i, vl_i \rangle$.

2.6.4.3 Planning Tasks

A MAP task is defined to find, collaboratively, a sequence of actions that transform the initial world state to a state satisfying a given goal condition. In CAMAP, a MAP **task** $\mathbb{M}$

is a 5-tuple $\langle \mathsf{AG}, \Psi, \Delta, \mathsf{A}, \mathsf{G} \rangle$ consisting of:

1. $\mathsf{AG} = \{\mathsf{Ag}_1 \dots \mathsf{Ag}_n\} \mid n = |AG|$ is a finite non-empty set of ambient agents with planning and argumentation capabilities. It is assumed that agents are fully cooperative.

2. Ψ is a set of values assigned to the state variables in V and represents the initial state of $\mathbb{M}$. Each $\Psi_{\mathsf{Ag}_i} \subseteq \Psi$ represents the partial view of the initial state of agent Ag_i such that $\Psi = \bigcup_{\forall \mathsf{Ag}_i \in \mathsf{AG}} \Psi_{\mathsf{Ag}_i}$ is a consistent set.

3. Δ is a finite set of (non-deterministic) defeasible rules. $\Delta_{\mathsf{Ag}_i} \subseteq \Delta$ is the set of rules known by agent Ag_i such that $\Delta = \bigcup_{\forall \mathsf{Ag}_i \in \mathsf{AG}} \Delta_{\mathsf{Ag}_i}$ is a set of possibly contradictory rules.

4. A is a finite set of deterministic planning actions. $\mathsf{A}_{\mathsf{Ag}_i} \subseteq \mathsf{A}$ is the set of actions known by agent Ag_i such that $\mathsf{A} = \bigcup_{\forall \mathsf{Ag}_i \in \mathsf{AG}} \mathsf{A}_{\mathsf{Ag}_i}$.

5. G is a set of global goals that denotes the needs of a user in an AmI environment. G is expressed as a set of pairs variable-value, indicating that each variable is expected to take on the corresponding value in the final state. Unlike the rest of elements, G is known by all of the ambient agents.

The group of agents AG involved in the resolution of the AmI application form a *planning team* since it is assumed that agents are fully cooperative, i.e., agents are not self-interested in CAMAP.

2.6.4.4 Arguments versus Actions

In what follows, we briefly introduce the theoretical framework underlying the context-aware system (for a more detailed description, see (23, 26, 34)). In the rest of the paper, we will use letters $\mathcal{A}, \mathcal{B}, \dots, \mathcal{Z}$ to denote arguments, and alpha symbols $\alpha_1, \alpha_2, \dots \alpha_n$, α_Ψ and α_G to represent actions. Both actions and arguments may be used to enforce some task goal in CAMAP. As illustrated in Figure 2.29, an **argument** $\mathcal{A}$ for $\langle v_i, vl_i \rangle$ proposed by an ambient agent Ag_1, is denoted as $\mathcal{A}^{\mathsf{Ag}_1} = (\{\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1})\}, \{\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})\})$, where $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \langle v_i, vl_i \rangle$ is the argument conclusion and $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})$ is a subset of defeasible rules such that $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1}) \subseteq \Delta_{\mathsf{Ag}_1}$. $\mathcal{A}^{\mathsf{Ag}_1}$ is consistent if there exists a defeasible derivation for $\langle v_i, vl_i \rangle$ from $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1}) \cup \mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})$, where $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ is the argument

base, the set of variable-value pairs that must be warranted in the agent's context information. The existence of an argument $\mathcal{A}^{\mathsf{Ag}_1}$ does not suffice to warrant its conclusion $\langle v_i, vl_i \rangle$, this depends on the interactions among arguments from different agents as we will see in Section 2.6.6.2. We semantically distinguish between **supporting arguments** (also known as argument steps) as the argument that agents specifically use to support some goal of the plan, and **attacking arguments** (also known as defeaters) which are only introduced to attack some argument step previously introduced in the plan by an ambient agent.

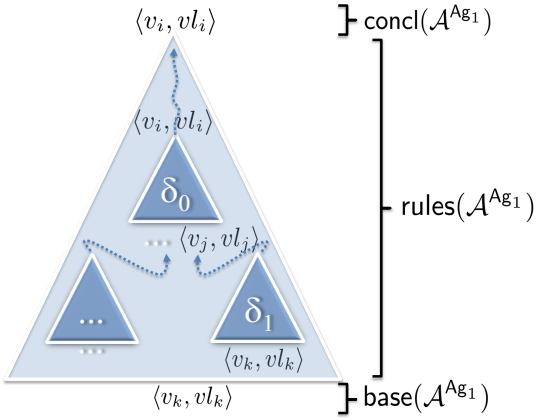

Figure 2.29: An argument $\mathcal{A}^{\mathsf{Ag}_1}$ for $\langle v_i, vl_i \rangle$ by using two defeasible rules, among others: $\delta_0 = \{\langle v_i, vl_i \rangle\} \multimap \{\langle v_j, vl_j \rangle\}$ and $\delta_1 = \{\langle v_j, vl_j \rangle\} \multimap \{\langle v_k, vl_k \rangle\}$, such that $v_i \neq v_j$, $v_j \neq v_k$ and $\{v_i, v_j, v_k\} \subseteq V_x$.

The difference between a conclusion $\langle v_i, vl_i \rangle$ derived by an argument or derived by an action is that in the case of a planning action the conclusion or effect is indisputable because it reflects a modification stated in the planning task modelling; however, the confirmation of a conclusion derived by an argument depends on the interaction with other attacking arguments. Put another way, planning actions are intended to express the *physics* of a domain so the effects of an action reflect the changes that will be produced in the world when the action is executed. However, an argument represents a belief inferred by an agent according to the knowledge and partial world view of such an agent so the belief may be invalidated if another agent puts forward an opposite conclusion.

2.6.4.5 Plans

In POP, a partial-order plan Π is a set of partially ordered actions (denoted by the relation $\prec$) which actually encodes multiple linear plans. More specifically, a **plan** Π is a tuple $\Pi = (\mathsf{A}(\Pi), \mathcal{AR}(\Pi), \mathsf{G}(\Pi), \mathcal{OC}(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi))$, where $\mathsf{A}(\Pi)$ denotes the set of action steps, $\mathcal{AR}(\Pi)$ represents the set of argument steps, $\mathsf{G}(\Pi)$ is the set of goals of the planning task (the user's needs), $\mathcal{OC}(\Pi)$ is a set of ordering constraints between actions in $\mathsf{A}(\Pi)$ (denoted by the relation $\prec$), and $\mathcal{CL}(\Pi)$ and $\mathcal{SL}(\Pi)$ represent the sets of causal and support links, respectively.

The initial state, Ψ, and goals, G, of a planning task, $\mathbb{M}$, are encoded as dummy actions $\{\alpha_\Psi \prec \alpha_\mathsf{G}\}$ where α_Ψ is also refereed to as the initial step of the plan and α_G to as the final step of the plan, with $\mathsf{X}(\alpha_\Psi) = \Psi$, $\mathsf{P}(\alpha_\mathsf{G}) = \mathsf{G}$, and $\mathsf{P}(\alpha_\Psi) = \mathsf{X}(\alpha_\mathsf{G}) = \emptyset$.

Let $\langle v_i, vl_i\rangle$ be an open goal (unsupported precondition) of some plan Π, motivated by some action step $\alpha_\mathsf{G} \in \mathsf{A}(\Pi)$, i.e., $\langle v_i, vl_i\rangle \in \mathsf{P}(\alpha_\mathsf{G})$; let $\langle v_k, vl_k\rangle$ be another open goal, motivated by some argument step $\mathcal{A}^{\mathsf{Ag}_1} \in \mathcal{AR}(\Pi)$, i.e., $\langle v_k, vl_k\rangle \in \mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ (Figure 2.43). The goal $\langle v_i, vl_i\rangle \in \mathsf{P}(\alpha_\mathsf{G})$ and must be supported by an argument, argument $\mathcal{A}^{\mathsf{Ag}_1}$ in Figure 2.43, which introduces the **support link** $(\mathcal{A}^{\mathsf{Ag}_1}, \langle v_i, vl_i\rangle, \alpha_\mathsf{G}) \in \mathcal{SL}(\Pi)$, where $\mathcal{SL}(\Pi) \subseteq \Delta \times \mathsf{G}(\Pi) \times \mathsf{A}$. In contrast, the goal $\langle v_k, vl_k\rangle$ must be supported by an action, α_1 in Figure 2.43, which introduces the **causal link** $(\alpha_1, \langle v_k, vl_k\rangle, \mathcal{A}^{\mathsf{Ag}_1}) \in \mathcal{CL}(\Pi)$, where $\mathcal{CL}(\Pi) \subseteq \mathsf{A} \times \mathsf{G}(\Pi) \times \Delta$. Specifically, in Figure 2.43, the real effects of α_1 ($\mathsf{X}(\alpha_1) = \langle v_i, vl_i\rangle$) are encoded through a supporting argument $\mathcal{A}^{\mathsf{Ag}_1}$ such that $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \langle v_i, vl_i\rangle$; $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1}) = \{\langle v_i, vl_i\rangle \multimap \langle v_k, vl_k\rangle\}$; and $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ is satisfied with $\langle v_k, vl_k\rangle$, a fictitious effect of the action α_1. The generation of $\langle v_k, vl_k\rangle$ is actually used as an indication that action α_1 has been executed while $\langle v_i, vl_i\rangle$, the real effects of α_1, are supported through an argument to let other agents discuss about the successful achievement of $\langle v_i, vl_i\rangle$ when α_1 is executed. This is an implicit way of accounting for the *qualification problem* (59), an open way for the rest of agents to be able to attack the supporting argument, as an indication that agents have their concerns regarding that the execution of α_1 will actually achieve $\langle v_i, vl_i\rangle$. However, it is important to note that supporting arguments are not only used as an intermediate step to derive the real effects of actions, but also as supporters to directly satisfy the open goal of an action step. In other words, open goals of action steps can be indirectly supported with the effects of another action through the use of an argument, or directly supported with the conclusion of an argument. This latter case means that the agent **supports the goal with a belief** of

its own so he believes **it is not necessary to introduce an action** for this purpose.

Let's suppose that an Emergency Medical Service (*EMS*) has been requested to assist injured people in an accident. CAMAP creates and runs a planning team, AG, to obtain a plan that satisfies the goal $\langle$ *at-EMS, accident-place* $\rangle$, i.e., a plan to assist and treat people involved in the accident. $\langle$ *at-EMS, accident-place* $\rangle$ is the goal of the planning task; *at-EMS* is a state variable and *accident-place* is the value the variable is expected to take on the final state. For this purpose, $\mathsf{Ag}_1 \in \mathsf{AG}$ constructs a partial plan that contains a supporting argument $\mathcal{A}^{\mathsf{Ag}_1}$ that derives $\langle$ *at-EMS, accident-place* $\rangle$, whose $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ is supported by the fictitious effect of an action α_1 that sends the ambulance *amb11* from the hospital *H1* to the accident place. Thus, $\langle$ *at-amb11, H1* $\rangle \in \mathsf{P}(\alpha_1)$ is the precondition of α_1, and the effects, $\mathsf{X}(\alpha_1)$, are supported by $\mathcal{A}^{\mathsf{Ag}_1}$ that derives the conclusion $\langle$ *at-EMS, accident-place* $\rangle$. At this point, the plan proposed by Ag_1 still has some open goals to be achieved, specifically the unsupported precondition $\langle$ *at-amb11, H1* $\rangle$ of α_1. The addition of new actions in the plan will eventually require preconditions such as having a doctor, a nurse or an ambulance driver available in *H1* to form the *EMS* that will be sent to the accident place.

Agents build **attacking arguments** for or against a supporting argument when they have beliefs (derived through their defeasible rules) that sustain or contradict the validity of the supporting argument. More specifically, $\mathcal{B}^{\mathsf{Ag}_2} = (\{\mathsf{concl}(\mathcal{B}^{\mathsf{Ag}_2})\}, \{\mathsf{rules}(\mathcal{B}^{\mathsf{Ag}_2})\})$ in Figure 2.43 is an attacking argument acting as a defeater of $\mathcal{A}^{\mathsf{Ag}_1}$ ($\mathcal{B}^{\mathsf{Ag}_2}$ attacks $\mathcal{A}^{\mathsf{Ag}_1}$), where $\mathsf{concl}(\mathcal{B}^{\mathsf{Ag}_2}) = \langle v_i, vl_{i'} \rangle \mid vl_i \neq vl_{i'}$, and v_i is a single-valued variable. This means that if an action α_1 is executed in a state in which $\langle v_s, vl_s \rangle$ holds (Figure 2.43 where $\langle v_s, vl_s \rangle \in \mathsf{base}(\mathcal{B}^{\mathsf{Ag}_2})$), then, Ag_2 has enough reasons to believe $\langle v_i, vl_{i'} \rangle$, which will cause α_1 to fail in the current context and thus not to produce its intended effect $\langle v_i, vl_i \rangle$. The blue triangle in Figure 2.43, $\mathcal{A}^{\mathsf{Ag}_1}$, represents an argument supporting the precondition $\langle v_i, vl_i \rangle$ of action α_{G}; and the yellow triangle, $\mathcal{B}^{\mathsf{Ag}_2}$, represents an argument attacking $\mathcal{A}^{\mathsf{Ag}_1}$. Rectangles represent action steps, i.e., actions that support the base of an argument step. The existence of $\mathcal{B}^{\mathsf{Ag}_2}$ means that agent Ag_2 has reasons to believe that ambulance *amb11* will not reach the accident place; this would occur, for instance, if according to Ag_2's knowledge *amb11* is not available. On the other hand, Ag_2 might also propose another plan with a supporting argument, $\mathcal{A}^{\mathsf{Ag}_2}$, which also derives $\langle$ *at-EMS, accident-place* $\rangle$ and whose $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_2})$ is supported by the effects of an action α_2 that sends another ambulance, *amb21*, from hospital *H2*, and which Ag_2 knows it is available.

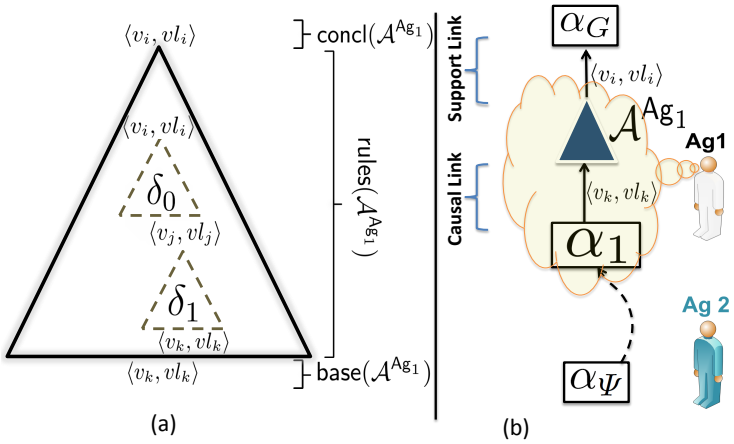

Figure 2.30: An example solving the qualification problem.

In our approach, goals must always be supported by the conclusion of an argument step, and the argument base must be satisfied by an action step (including the initial step). This way, a typical causal link in POP is now replaced by a causal link and a support link. Note that this representation allows us to implicitly address *the qualification problem* as every precondition of a planning action is now supported by an argument step rather than directly by an action effect. The default approach in traditional planning to the qualification problem is to assume the world will behave as expected, that no unexpected circumstances may at any time prevent the successful performance of an action and, therefore, the achievement of the desired effects. In CAMAP, accounting for the unexpected circumstances that may happen in the context where the plan is being constructed is done via argumentation, allowing agents to build an attacking argument against the argument that supports the effect of an action α. This way, an attack to an argument stands for any unexpected contingency that would prevent α from being successfully executed, i.e., for any precondition not listed in $\mathsf{P}(\alpha)$ that would prevent the execution of α from having its intended effects. Ambient agents are thus enabled to attack any step of the plan under construction.

Finally, we distinguish between the notions of *threat* and *attack* that may arise during the plan construction: a threat is a conflict that arises during the planning process, an interference between a support link and an argument step; an attack is a conflict that

arises during the argumentative process, an interference between a support link and an attacking argument. This is referred to as planning threats (threats) versus argumentation threats (attacks) in (34).

2.6.5 Overview of the Ambient Intelligence Application Scenario

This section provides a brief overview of the AmI application upon which the framework CAMAP will be applied. We highlight the fact that CAMAP is a domain-independent planning system but in this paper we apply CAMAP to a specific application of AmI in the field of health-care.

The emerging advances in pervasive computing technologies hold great potential for improving people's quality of life. One of the most promising area of applications of these technologies is home health-care (90, 91, 106). In recent years, remote monitoring of patients (hospitalized at home) in real-time via wearable health monitoring devices has become a special focus of interest (107). For instance, monitoring people prone to suffer from heart diseases gives rise to a task of periodically controlling the patient's heart in order to prevent a premature death. Here, we assume that the patient's home is equipped with appropriate technologies to create the AmI environment. The patient is monitored with a bracelet, which collects the patient's physical activity and wirelessly transmits it to a device responsible for monitoring patient's heart rate.

Health-care applications are more concerned with the wearable technology required for patient monitoring (107), but the real problem arises when an anomaly is detected in the monitor readings, e.g., an extremely low level of the patient's physical activity which may end up in a heart attack. These situations claim the construction of a plan that: i) sends a health service to the place where the patient is, ii) assists the patient, and iii) moves the patient to a hospital, in case this is necessary. The construction of a plan to tackle this type of situations is still a challenge for health-care systems in AmI. In fact, to the best of our knowledge, there are no works that apply an automated planning process in context-aware environments with multiple intervening entities. In this paper, we propose to execute CAMAP for assisting a patient whose monitor readings report some disfunctions.

2.6.5.1 Modeling the Health-Scenario with a Planning Language

CAMAP uses a language based on the latest version of PDDL (Planning Domain Definition Language), PDDL3.1 (78), which was introduced in the context of the 2008 International Planning Competition. Unlike its predecessors, that model a planning domain through logical predicates, PDDL3.1 also incorporates state variables by adding object fluents that map a tuple of objects to an object of the problem.

PDDL, the most popular language for modeling planning tasks, allows for the specification of the components of a planning task, namely type of objects specified in a structure called **`:types`**; objects specified in a structure called **`:objects`**; predicates specified in a structure called **`:predicates`**; single-valued state variables specified in a structure called **`:functions`**; planning actions specified in a structure called **`:action`**; the initial state specified in a structure called **`:init`** and goal state of the task specified in a structure called **`:global-goal`**. Additionally, we have extended PDDL3.1 in order to introduce some functionalities required in a multi-agent planning task. Specifically, we incorporate new structures in the language to define:

(a) The multi-agent features of the planning task; this requires the specification of multiple planning problems, one per agent, defining the abilities, initial state and planning context of each agent. Since information of the planning task is distributed across agents, we have also created specific structures to define the information that agents will exchange between each other during the planning process.

(b) The set of defeasible rules of the agents is defined through the additional structure **`:def-rule`**.

In this section, we focus on the definition of the following elements: AG, the ambient agents, V, the state variables, Δ, the defeasible rules, A, the planning actions, O, the planning objects, Ψ, the initial state, and G, the goal state.

Object Types and Ambient Agents. Planning objects are the basic entities of a MAP task. In PDDL, it is possible to define object types and create a hierarchy of types. As shown in Listing 2.5, we define the following principal object types:

- `hospital`, is an object type that represents the existing infrastructure of a hospital. By `hospital` we refer to all the components that constitute a hospital entity, i.e., hospital rooms, units of the hospital, staff working in the hospital, etc.

- the type `emergency-medical-transportation` comprises the vehicles that hospitals utilize for assisting an emergency; e.g., `ambulance`, `helicopter`, etc. In turn, ambulances are divided into ALS (Advanced Life Support) and BLS (Basic Life Support) depending on the equipment of the ambulance.
- `patient`, represents the injured patients. It includes those who are hospitalized in hospital and those who are hospitalized at home.
- `patient-disease`, represents different types of diseases. It is divided into seven types of diseases.
- `address` is a type that represents the address of the locations.
- `density` is a type that represents the density of traffic.
- `distance` is a type that represents the distance between two places.

In our health application, we have the following types of ambient agents:

(a) Transport agents, whose main function is to guide the ambulance/helicopter to follow the best route to reach the patient's home.

(b) Communication agents in charge of using telecommunication devices such as a cell telephone to call the emergency services.

(c) Assistant agents, who are responsible for controlling an automated external defibrillator, an activity tracking device, a position tracking device, etc. to interact with both the environment and the user.

```
(:types
  [hospital emergency-medical-transportation] - object
  [patient patient-disease address density distance] - object
  [patient-home patient-hospital] - patient
  [hospital-name hospital-tower hospital-floor hospital-unit] - hospital
  [hospital-bed hospital-room EMS-team hospital-staff] - hospital
  [doctor nurse driver first-aid-assistants security-guards] - hospital-staff
  [critical-patient-unit deceased-patient-unit other-unit] - hospital-unit
  [intensive-care neonatology reanimation burnt] - critical-patient-unit
  [ambulance helicopter] - emergency-medical-transportation
  [BLS-ambulance ALS-ambulance] - ambulance
  [BLS-EMS-team ALS-EMS-team] - EMS-team
  [mentally-illness chronic-illness terminal-illness] - patient-disease
  [short-illness long-illness minor-illness] - patient-disease)
```

Listing 2.5: Types of objects.

In the particular scenario case we present in this paper, we deal with only one transport agent (Ag_1), one communication agent (Ag_2) and one assistant agent (Ag_3).

State Variables. In PDDL3.1, state variables are specified as functions with any number of parameters. Listing 2.6 shows the definition of some of the functions we use in our health-care scenario. For instance, `(pos ?a - ambulance) - address` specifies a state variable that represents the position of an ambulance, and the value of this variable is an object of type `address`. And `(deviceTraffic ?ad1 - address ?ad2 - address) - density` is a state variable that returns the traffic density between two given addresses. As we can see in Listing 2.6, state variables are defined as functions where the first element is the function name and the rest of elements are typed parameters.

```
(:functions
  (pos ?a - ambulance) - address
  (pos ?m - EMS-team) - address
  (pos ?p - patient) - address
  (location ?h - hospital-name) - address
  (moved ?a - ambulance ?ad - address) - address
  (moved ?m - EMS-team ?ad - address) - address
  (deviceTraffic ?ad1 - address ?ad2 - address) - density
  (deviceDistance ?ad1 - address ?ad2 - address) - distance
  ...)
```

Listing 2.6: Single-valued variables represented as functions.

Additionally, particular pieces of information are modelled with predicates as shown in Listing 2.7. For instance, `(carpoolLaneBetween ?ad1 - address ?ad2 - address)` is a predicate that indicates whether or not there is a carpool lane (an express lane) that emergency vehicles can use to quickly reach a location.

```
(:predicates
  (trafficJamBetween ?ad1 - address ?ad2 - address)
  (carpoolLaneBetween ?ad1 - address ?ad2 - address)
  (isFarFrom ?ad1 - address ?ad2 - address)
  (assistingThePatient ?p - patient)
  (toBeAssisted ?p - patient)
  ...)
```

Listing 2.7: Predicates.

Finally, CAMAP language also supports multi-functions, i.e., state variables that can take on several values (multi-valued variables). This is done trough the additional structure `:multi-functions`. Listing 2.8 shows one of the multi-valued variables denoting the sanitary coverage area of a hospital. The area covered by each hospital includes several addresses although one same address may belong to different sanitary coverage areas.

```
(:multi-functions
  (sanitaryCoverage ?h - hospital) - address
  ...)
```

Listing 2.8: Multi-valued variables represented as multi-functions.

Defeasible Rules and Actions. Listing 2.9 and listing 2.10 show the structure of a planning action and a defeasible rule of agent Ag_1, respectively. As we can see in the listings, the symbol `=` is used to check whether the value of a single-valued variable matches a specific value, while `member` is used to test if a multi-valued variable contains a specific value. Both `=` and `member` are the comparison operators in CAMAP for single-valued and multi-valued variables, respectively. On the other hand, `assign` is used to assign a value to a state variable, either in an action rule or defeasible rule. For the ease of specification, we will continue using the notation variable-value pair $\langle v_i, vl_i \rangle$ to refer to preconditions and effects of actions as well as the body and head of defeasible rules within the text.

Action **`moving-BLS-medical-assistance`** in Listing 2.9 represents the movement of a BLS-ambulance from one location to another. The effects of this action are actually fictitious conclusions that are used for a defeasible rule to derive the real effects of the action, i.e., to have the ambulance and the *EMS* team at the patient's home and the patient being assisted. Recall that the effects of an action are used to support the base of an argument, which in turn is used as a mechanism to allow agents dispute about the successful execution of an action (see our interpretation of the *qualification problem* in Section 2.6.4.5). Listing 2.10 shows the defeasible rule **`moved-BLS-medical-assistance`** that derives the real effects of the action **`moving-BLS-medical-assistance`** and whose body matches the fictitious conclusions of that action.

Agents have different capabilities according to their role so they will contribute with different actions in the plan construction. On the other hand, the beliefs of an agent are the derivations of its defeasible rules, and these may relate to any aspect of the context information. That is, agents can make assumptions on the current status of the application regarding any issue of the AmI environment.

```
(:action moving-BLS-medical-assistance
  :parameters (?a - BLS-ambulance ?h - hospital-name ?ad1 - address ?ad2 - address
  ?m - BLS-EMS-team ?p - patient-home)
  :precondition (and (member (sanitaryCoverage ?h) ?ad2)
                     (= (pos ?a) ?ad1)
                     (= (pos ?m) ?ad1)
                     (= (pos ?p) ?ad2)
                     (= (location ?h) ?ad1))
  :effect (and (assign (moved ?a ?ad1) ?ad2)
               (assign (moved ?m ?ad1) ?ad2)
               (toBeAssisted ?p)))
```

Listing 2.9: An action for moving an ambulance from a location to another.

```
(:def-rule moved-BLS-medical-assistance
  :parameters(?a - BLS-ambulance ?ad1 - address ?ad2 - address ?m - BLS-EMS-team
  ?p - patient-home)
  :body (and (= (moved ?a ?ad1) ?ad2)
             (= (moved ?m ?ad1) ?ad2)
             (toBeAssisted ?p))
  :head (and (assistingThePatient ?p)
             (assign (pos ?a) ?ad2)
             (assign (pos ?m) ?ad2)))
```

Listing 2.10: The body of the defeasible rule matches the effects of the action moving-BLS-medical-assistance to deal with the qualification problem.

Objects, Initial State and Goals. Listing 2.5 displays the object types of our healthcare scenario. Particularly, in our problem we model three hospitals in a city {`H1, H2, H3`} (Listing 2.11). Each hospital disposes of two ambulances out of 6 ambulances available in the problem {`amb11`, `amb12`, `amb21`, `amb22`, `amb31`, `amb32`}. One ambulance is equipped with an ALS equipment, and the other is equipped with a BLS equipment. Each hospital has also one emergency helicopter out of three helicopters available in the problem {`he1`, `he2`, `he3`}. Moreover, there are two *EMS* teams on call in each hospital from

the set {t11, t12, t21, t22, t31, t32}: one handles the ALS emergency equipment, and is formed by an ambulance driver, a nurse and a physician; the other handles the BLS equipment and is formed by an ambulance driver and a nursing assistant.

```
(:objects
  [amb11 amb21 amb31] - BLS-ambulance
  [amb12 amb22 amb32] - ALS-ambulance
  [he1 he2 he3] - helicopter
  [H1 H2 H3] - hospital-name
  [t11 t21 t31] - BLS-EMS-team
  [t12 t22 t33] - ALS-EMS-team
  [high medium low] - density
  [long normal short] - distance
  [p1 p2 ... p50] - patient-home
  [aH1 aH2 aH3 pH1 pH2 ... pH50] - address
  ...)
```

Listing 2.11: The objects that encodes the elements of the planning task.

In CAMAP, as in many MAP systems, agents do not have a complete view of the world and, consequently, some details of the problem may be unknown to them. In this sense, unlike most planning representations that adopt the closed-world assumption, here we associate each piece of information with three possible truth values: *true*, *false* and *unknown*. Particularly, if a single-valued variable v_i is assigned the value vl_i then the variable-value pair $\langle v_i, vl_i \rangle$ is true, and the pair $\langle v_i, vl_j \rangle \mid vl_j \neq vl_i$ is false because the variable is not assigned the value vl_j. In case the variable has not yet been assigned a value, any pair $\langle v_i, vl_i \rangle$ is unknown. Since we now allow for three possible logic states of the information, we have to explicitly represent the true and false information, leaving the unknown state to the information that does not appear explicitly in the representation of a world state. Specifically, the initial state of an ambient agent in CAMAP is composed of: (i) the positive and negative values of the single-valued variables in the structure **:functions**; (ii) one or more values for the multi-valued variables specified in the structure **:multi-functions**; and, (iii) the positive and negative facts specified in the structure **:predicates**.

For instance, in our health-care domain, the state variable *pos-p1* indicates the location of a patient p1, which is assigned a value of type address; otherwise, while the variable is not assigned a value, all possible $\langle$ *pos-p1,* $vl_{pos-p1} \rangle$ pairs are unknown. Values pH1, pH2, etc. denote the home address of patients hospitalized at home whereas aH1, aH2, etc. are addresses of the hospitals in which patients are hospitalized (this is the case, for instance,

of patients with a serious health deterioration). Listing 2.12 shows a partial description of the information that the transport agent, Ag_2, knows about the initial state (Ψ_2). Ψ_2 represents a situation in which patient `p1` is not being assisted yet ($\langle$ *assistingThePatient-p1, false* $\rangle$), there is a high traffic density between the hospital (denoted by address `aH1`), and the patient's home (denoted by address `pH1`), and the two locations are far away from each other (long distance between them). Consequently, the pairs $\langle$ *deviceTraffic-aH1-pH1, medium* $\rangle$ and $\langle$ *deviceTraffic-aH1-pH1, low* $\rangle$ are false. The symbol = is used in Listing 2.12 to assign a value to a variable according to PDDL3.1 specifications.

```
(:init
  (= (pos p1) pH1)
  (= (location H1) aH1)
  (not (assistingThePatient p1))
  (= (deviceTraffic aH1 pH1) high)
  (not (deviceTraffic aH1 pH1) medium)
  (not (deviceTraffic aH1 pH1) low)
  (= (deviceDistance aH1 pH1) long)
  ...)
```

Listing 2.12: Initial state of agent Ag_2.

Let's assume that the health of patient `p1`, who is hospitalized at home ($\langle$ *pos-p1, pH1* $\rangle$) and monitored from a hospital, suddenly worsens considerably. CAMAP is then required to build a plan to assist the patient `p1`; this is represented by defining the goal $\langle$ *assistingThePatient-p1, true* $\rangle$ as shown in Listing 2.13.

```
(:global-goals
    (assistingThePatient p1)
    ...)
```

Listing 2.13: Global goals.

2.6.6 Context-Aware Multi-Agent Planning Protocol

In this section, we outline the CAMAP protocol that works in three steps; a *planning stage*, an *argumentation stage* and a *selection stage*.

Given a set of global goals, G, representing the requirements of an AmI application, agents build their own partial view of the planning task $\mathbb{M}$ so that they will contribute differently to the construction of the joint solution plan. The CAMAP protocol starts

with a plan, $\Pi_0 = \{\alpha_\Psi \prec \alpha_\mathsf{G}\}$, that is initially empty and searches through a space of possible plans. A successful search terminates with a solution plan, i.e., a plan in which all action step preconditions are necessarily true. The search space is characterized as a POP tree, $\mathbb{T}$, where each node corresponds to a plan and each arc corresponds to a plan transformation (108). We denote by $\mathsf{OpNodes}(\mathbb{T})$ the set of leaf nodes of $\mathbb{T}$, i.e., the candidate nodes that have not been expanded yet. Initially, $\mathsf{OpNodes}(\mathbb{T}) = \{\Pi_0\}$. The plan Π selected from $\mathsf{OpNodes}(\mathbb{T})$ at each time is called the base plan, and this is the plan over which the agents will create their refinements.

The overall idea of CAMAP protocol is to collaboratively and progressively refine the base plan until it becomes a solution plan. Given the base plan Π, the first step is to select an open goal $\Phi \in \mathsf{G}(\Pi)$ of the planning task (Goal Selection Process in Figure 2.31). Then it comes the planning stage (Plan Proposal Process in Figure 2.31) where agents put forward and exchange different partial-order plans that would potentially solve Φ. Following, agents become involved in an argumentative dialogue (Plan Evaluation Process in Figure 2.31) in which they expose their arguments for or against each of the plan proposals. This Plan Evaluation Process performs a *warranty procedure* to determine which proposals do not receive attacks, or otherwise, the received attacks do not succeed. Subsequently, ambient agents reach an agreement about the next base plan $\Pi \in \mathsf{OpNodes}(\mathbb{T})$ to be refined (Plan Selection Process in Figure 2.31) and they continue the search exploration. The process is repeated until a solution plan is found.

2.6.6.1 **Plan** Proposal Process

Plan refinements, or simply refinements, denote the plan proposals put forward by ambient agents to solve a selected open goal Φ of a base plan $\Pi \in \mathsf{OpNodes}(\mathbb{T})$. This stage follows a process similarly to a plan-space planning process that builds a POP tree, except that each refinement or successor of Π can be generated now by a different agent. Another distinguishing characteristic of CAMAP is that the nodes also contain argument steps, as explained in Section 2.6.4.5, to support action preconditions; this argument structure of the plans will be later used in the Plan Evaluation Process. We denote by $\mathsf{Ref}(\Pi, \Phi)$ the set of refinement proposed by the agents to solve an open goal Φ of a base plan Π.

The Plan Proposal Process finishes when all agents come up with their plan proposals at their turn and these are communicated to the rest of agents. Then, agents update their set of actions with the information appearing in the refinements proposed by the other

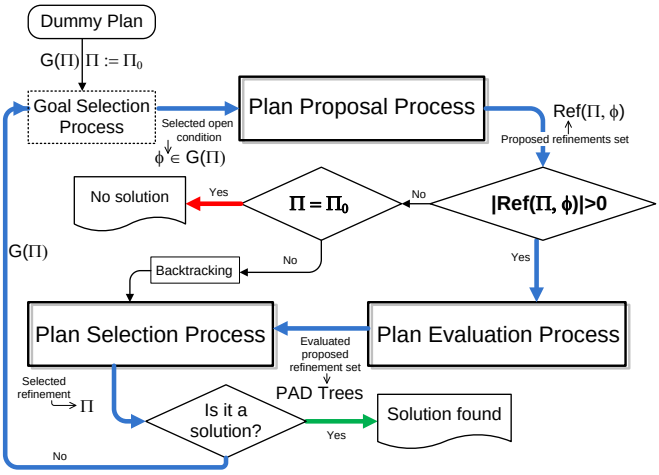

Figure 2.31: Overview of the CAMAP protocol.

agents, and the elements in the set $\mathsf{Ref}(\Pi, \Phi)$ are added to $\mathsf{OpNodes}(\mathbb{T})$.

Let's suppose that ambient agents are asked to solve the open goal $\mathsf{P}(\alpha_\mathsf{G}) = \langle$ *assistingThePatient-p1, true* $\rangle$ in our AmI scenario described in Section 2.6.5. Agent Ag_1, the transport agent, generates at least 6 refinement plans (3 hospitals x 2 ambulances) by using the planning action and defeasible rule shown in Listings 2.9 and 2.10, respectively, among others. As shown Figure 2.32(a), one of the refinements proposed by agent Ag_1 is $\Pi_r \in \mathsf{Ref}(\Pi, \mathsf{P}(\alpha_\mathsf{G}))$ such that $\mathcal{OC}(\Pi_r) = \{\alpha_\Psi \prec \alpha_1; \alpha_1 \prec \mathcal{A}^{\mathsf{Ag}_1}; \mathcal{A}^{\mathsf{Ag}_1} \prec \alpha_\mathsf{G}\}$, where:

- $\mathcal{A}^{\mathsf{Ag}_1}$ is an argument built by using the defeasible rule **`moved-BLS-medical-assistance`** such that:
  - $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \{\langle$ *assistingThePatient-p1, true* $\rangle, \langle$ *pos-t11, pH1* $\rangle, \langle$ *at-amb11, pH1* $\rangle\} \supseteq \mathsf{P}(\alpha_\mathsf{G})$.
  - $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1}) = \{\langle$ *moved-amb11-aH1, pH1* $\rangle, \langle$ *moved-t11-aH1, pH1* $\rangle, \langle$ *toBeAssisted-p1, true* $\rangle\}$.
- α_1 is a ground action out of **`moving-BLS-medical-assistance`** such that:

- $\mathsf{X}(\alpha_1) = \{\langle$ *moved-amb11-aH1, pH1* $\rangle, \langle$ *moved-t11-aH1, pH1* $\rangle, \langle$ *toBeAssisted-p1, true* $\rangle\}$ that matches $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$.
- $\mathsf{P}(\alpha_1) = \{\langle$ *sanitaryConverage-H1, {pH1}* $\rangle, \langle$ *at-amb11, aH1* $\rangle, \langle$ *pos-t11, aH1* $\rangle, \langle$ *pos-p1, pH1* $\rangle, \langle$ *pos-H1, aH1* $\rangle\}$.

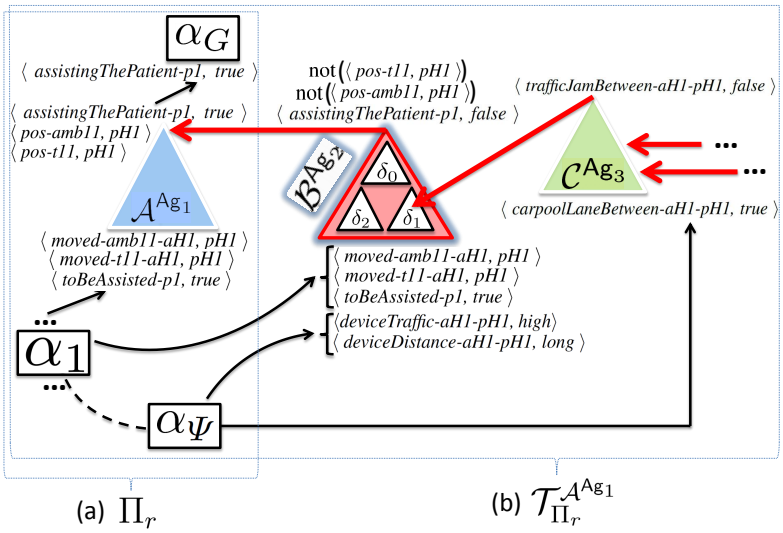

Figure 2.32: Examples of the (a) Plan Proposal Process, and (b) Plan Evaluation Process.

The conclusion of argument $\mathcal{A}^{\mathsf{Ag}_1}$ derives the real effects of the action α_1, and $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ is supported by the fictitious effects of the action α_1, $\mathsf{X}(\alpha_1)$ (Figure 2.32(a)).

In most MAP systems (29, 31, 103), open goals are supported by the inclusion of an action whose effects match the open goals. However, as we said in Section 2.6.4.5, CAMAP allows agents to directly enforce action preconditions with their beliefs, meaning that agents believe these **preconditions can be supported with their context inferences**, without having to consider taking action to meet them. For example, suppose that an agent extracts information from a car position tracking device that there is already an ambulance available at the patient's home. In this case, the agent will meet the open goal ⟨ *assisting-ThePatient-p1, true* ⟩ with an argument that reflects its belief that an ambulance is already at the patient's. As the rest of agents do not own this information, their proposals would claim the inclusion of a planning action that moves an ambulance or helicopter to the

indicated place.

2.6.6.2 **Plan** Evaluation Process

At the Plan Evaluation Process, agents become engaged in a series of argumentative dialogues aimed at evaluating the guarantee of a successful execution of a plan proposal. Specifically, argumentation occurs when agents believe it is likely that unexpected circumstances will prevent the action from achieving its intended effects or when they are against a belief that has been used to meet some open goal. Agents will build their arguments on the basis of their context information and inferences, which may differ to each other. Therefore, agents may not agree on the evaluation of a plan proposal at some point during the plan construction.

The Plan Evaluation Process generates as many argumentative dialogues as argument steps are present in a refinement. An argumentative dialogue is an exchange of arguments for or against the fulfillment of a particular argument step $\mathcal{A}^{\mathsf{Ag}_i} \in \mathcal{AR}(\Pi_r)$ such that $\Pi_r \in \mathsf{OpNodes}(\mathbb{T})$ and $\mathsf{Ag}_i \in \mathsf{AG}$.

In CAMAP, an argumentative dialogue is encoded as a tree structure, called Plan Argument Dialogue (PAD) tree. Specifically, the PAD tree to evaluate an argument $\mathcal{A}^{\mathsf{Ag}_i}$ in plan Π_r, is denoted as $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$. The nodes of a PAD tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$ are labeled with an argument that attacks the argument represented by its parent node and whose bases are supported in the plan Π_r. More specifically:

1. The root node of the tree is labeled with $[\mathcal{A}^{\mathsf{Ag}_i}]$ such that $\mathcal{A}^{\mathsf{Ag}_i} \in \mathcal{AR}(\Pi_r)$.

2. A child node $[\mathcal{B}^{\mathsf{Ag}_j}]$ of the parent node $[\mathcal{A}^{\mathsf{Ag}_i}]$ represents an attacking argument against the argument $\mathcal{A}^{\mathsf{Ag}_i}$ in plan Π_r, i.e., $\mathcal{B}^{\mathsf{Ag}_j}$ is a defeater of $\mathcal{A}^{\mathsf{Ag}_i}$. Consequently, each child node of $[\mathcal{A}^{\mathsf{Ag}_i}]$ stands for a defeater of the root argument $[\mathcal{A}^{\mathsf{Ag}_i}]$,

3. A child node $[\mathcal{C}^{\mathsf{Ag}_z}]$ of the parent node $[\mathcal{B}^{\mathsf{Ag}_j}]$ indicates an attack to argument $\mathcal{B}^{\mathsf{Ag}_j}$, so this new node is actually a supporter of the root argument $\mathcal{A}^{\mathsf{Ag}_i}$.

And so on for the rest of nodes. The leaves of the PAD tree correspond to undefeated arguments. Informally, we might see a PAD tree for an argument step $\mathcal{A}^{\mathsf{Ag}_i}$ as generating a *dialectical tree* for $\mathcal{A}^{\mathsf{Ag}_i}$ in order to determine whether argument $\mathcal{A}^{\mathsf{Ag}_i}$ is warranted or not (24). Unlike the dialectical trees of a general argumentative process, the nodes in our PAD tree are contextualized within a plan. Every linear path from the root to a leaf

corresponds to one different acceptable *argumentation line*. Circular argumentation (also known as fallacious argumentation) is avoided by applying both conditions from (24): no argument can be reintroduced in the same argumentation line and argument concordance must be guaranteed.

Following, we will explain an example of the Plan Evaluation Process to evaluate the argument step $\mathcal{A}^{\mathsf{Ag}_1}$ in plan Π_r of Section 2.6.6.1. This is graphically shown in Figure 2.32(b). Ag_1 starts the process by sending the PAD tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$, which only contains the root node $[\mathcal{A}^{\mathsf{Ag}_1}]$, to the rest of ambient agents. When $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ is received by ambient agent Ag_2, it reads the traffic density between the hospital location `aH1` and the patient's location `pH1` from a smart device connected to the AmI system, and the reading returns a high traffic density between the two locations. In addition, Ag_2 knows the two locations are far away from each other thanks to the a web mapping service as Google Maps. Both informations, which are unknown to ambient agent Ag_1, may be a reason for Ag_2 to believe that an ambulance, initially located at the hospital `aH1` will not arrive at `pH1` in time for assisting the patient `p1`. More specifically, Ag_2 puts forward an attacking argument $\mathcal{B}^{\mathsf{Ag}_2}$ =({⟨ *assistingThePatient-p1, false* ⟩, *not*(⟨ *pos-amb11, pH1* ⟩), *not*(⟨ *pos-t11, pH1* ⟩)}, {δ_0; δ_1; δ_2}) that attacks $\mathcal{A}^{\mathsf{Ag}_1}$, as shown in Figure 2.32(b); this attack is derived from the defeasible rules in Listing 2.14 where:

- δ_0 =(*and* ⟨ *assistingThePatient-p1, false* ⟩ *not*(⟨ *pos-amb11, pH1* ⟩) *not*(⟨ *pos-t11, pH1* ⟩)) $\prec$ (*and* ⟨ *moved-amb11-aH1, pH1* ⟩ ⟨ *moved-t11-aH1, pH1* ⟩ ⟨ *toBeAssisted-p1, true* ⟩ ⟨ *trafficJamBetween-aH1-pH1, true* ⟩ ⟨*isFarFrom-aH1-pH1, true*⟩).
- δ_1 =⟨ *trafficJamBetween-aH1-pH1, true* ⟩ $\prec$⟨*deviceTraffic-aH1-pH1, high*⟩.
- δ_2 =⟨ *isFarFrom-aH1-pH1, true* ⟩ $\prec$ ⟨ *deviceDistance-aH1-pH1, long* ⟩.

```
(:def-rule moved-BLS-medical-assistance-denied
  :parameters(?a - BLS-ambulance ?ad1 - address ?ad2 - address ?m - BLS-EMS-team
  ?p - patient-home)
  :body (and (= (moved ?a ?ad1) ?ad2)
             (= (moved ?m ?ad1) ?ad2)
             (toBeAssisted ?p)
             (trafficJamBetween ?ad1 ?ad2)
             (isFarFrom ?ad1 ?ad2))
  :head (and (not (assistingThePatient ?p))
             (not (pos ?a) ?ad2)
             (not (pos ?m) ?ad2)))
(:def-rule traffic-jam
```

```
  :parameters(?ad1 - address ?ad2 - address)
  :body (= (deviceTraffic ?ad1 ?ad2) high)
  :head (trafficJamBetween ?ad1 ?ad2))
(:def-rule is-far-away
  :parameters(?ad1 - address ?ad2 - address)
  :body (= (deviceDistance ?ad1 ?ad2) long)
  :head (isFarFrom ?ad1 ?ad2))
```

Listing 2.14: Defeasible rules of agent Ag_2 for representing situations in which the ambulance may not arrive on time.

Assuming that $\{\langle$ *deviceTraffic-aH1-pH1, high* $\rangle$, $\langle$ *deviceDistance-aH1-pH1, long* $\rangle\} \subseteq \Psi_{\mathsf{Ag}_2}$, as shown in Listing 2.12 of Section 2.6.5.1, Ag_2 sends $[\mathcal{B}^{\mathsf{Ag}_2}]$ to Ag_1, who manages the argumentative dialog. The first argumentative round ends here since no other agent disputes $[\mathcal{A}^{\mathsf{Ag}_1}]$.

In the next round of the dialogue, Ag_1 updates the PAD tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ with $[\mathcal{B}^{\mathsf{Ag}_2}]$ and sends it to the rest of agents. $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ is received by ambient agent Ag_3, but it knows about the existence of a carpool lane between `aH1` and `pH1`, which is a reason for Ag_3 to believe that the ambulance `amb11` can skip the traffic congestion on the way to reach the patient's home (Listing 2.15). Thus, Ag_3 builds a new attacking argument $\mathcal{C}^{\mathsf{Ag}_3} = (\{\langle$ *trafficJamBetween-aH1-pH1, false* $\rangle\}, \{\langle$ *trafficJamBetween-aH1-pH1, false* $\rangle \prec \langle$ *carpoolLaneBetween-aH1-pH1, true* $\rangle\})$ that defeats $\mathcal{B}^{\mathsf{Ag}_2}$, such that $\langle$ *carpoolLaneBetween-aH1-pH1, true* $\rangle \in \Psi_{\mathsf{Ag}_3}$ (Figure 2.32(b)). Ag_3 sends $[\mathcal{C}^{\mathsf{Ag}_3}]$ to agent Ag_1.

```
(:def-rule carpool-lane
  :parameters(?ad1 - address ?ad2 - address)
  :body (carpool-lane-between ?ad1 ?ad2)
  :head (not (traffic-jam-between ?ad1 ?ad2)))
```

Listing 2.15: Defeasible rule used by Ag_3 that derives the possibility of avoiding a traffic congestion situation if a carpool lane exits between two locations.

In another argumentation line against $\mathcal{A}^{\mathsf{Ag}_1}$, Ag_2 might argue that patient `p1` is in a critical state according to its context information. In this case, Ag_2 has reasons to believe that `p1` should be attended by a physician who could immediately diagnose and treat the patient. However, since the BLS medical equipment is only formed by an ambulance driver and a nursing assistant (Section 2.6.5.1), Ag_2 builds a new attacking argument $\mathcal{D}^{\mathsf{Ag}_2}$ against $\mathcal{A}^{\mathsf{Ag}_1}$ such that $\mathsf{concl}(\mathcal{D}^{\mathsf{Ag}_2}) = \langle$ *assistingThePatient-p1, false* $\rangle$. Ag_1 will

update the PAD tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ with the new received defeaters and will send it back again to the rest of ambient agents.

The dialogue process continues until none of the ambient agents has more arguments against any of the arguments in the PAD tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$. Then, CAMAP invokes the *warrant procedure* in order to check whether the argument $\mathcal{A}^{\mathsf{Ag}_1}$ is defeated or undefeated: label with a *U* (for *undefeated*) each terminal node in the PAD tree (i.e., each argument with no defeaters at all). Then, in a bottom-up fashion, CAMAP labels a node with: *U* if each of its successors is labeled with a *D*; and *D* (for defeated) otherwise. Note that the evaluation of $\mathcal{A}^{\mathsf{Ag}_1}$ is a key issue in the overall process as the patient's stabilization will most likely depend on a correct plan execution, specifically on the timely arrival of an ambulance.

A refinement plan is labeled as an **undefeated refinement plan** if the root node of the PAD tree of each argument in the plan results undefeated. Otherwise, the plan is provisionally labeled as a **defeated refinement plan** in the POP tree. Undefeated plans are obviously preferred over defeated plans as they represent a plan with no expectation of failures according to the ambient agents. Nevertheless, defeated plans are maintained in the POP tree as their arguments may become undefeated as long as the problem evolves and information changes. Finally, each ambient agent updates its initial facts and defeasible rules with the information exchanged during the argumentative process.

2.6.6.3 Plan and Goal Selection Process

The aim of the Plan Selection Process is to select the *best* plan Π from the list $\mathsf{OpNodes}(\mathbb{T})$ and then choose a goal to solve from this plan (Goal Selection Process). Once this is done, the CAMAP protocol starts all over again with the Plan Proposal Process.

In order to select the next best plan, we consider a compromise between different parameters: maximizing the likelihood of a successful execution of the solution plan; and, minimizing both the computational overhead and the total time of the search. The application of the first parameter discards the plans evaluated as defeated in the Plan Evaluation Process. With respect to the second parameter, we apply a heuristic function over the undefeated plans resulting from the first filtering. We use two of the most popular heuristics in planning: SUM and MAX heuristics (79). The SUM heuristic estimates the cost of a plan as the sum of the cost of the pending open goals in the plan whereas the MAX heuristic estimates the cost of the plan as the cost of the most costly open goal in the plan. Plans whose heuristic estimation is above a certain threshold are discarded from

consideration.

On the other hand, the aim of the Goal Selection Process is to choose an open goal Φ from the selected base plan Π. Specifically, Φ can be a goal of the planning task $\mathbb{M}$ or a subgoal that appears in Π as a result of the planning process. Among all the pending open goals in $\mathsf{G}(\Pi)$, we apply a heuristic function that selects the most costly open goal. Afterwards, CAMAP proceeds with the Plan Proposal Process unless Π becomes a solution plan, in which case CAMAP stops.

2.6.7 Experimental Evaluation

The purpose of this section is to compare CAMAP with a Traditional Multi-Agent Planning (TMAP) system with no argumentation mechanism for reasoning about the context information. The final objective is to analyze the benefits and limitations of both approaches.

2.6.7.1 Experimental Settings

We carried out several experiments considering three different levels of difficulty of the planning problems: small-size problems (composed of 8 ground actions[1] and 50 ground defeasible rules), medium-size problems (composed of 16 ground actions and 100 ground defeasible rules) and large-size problems (composed of 24 ground actions and 150 ground defeasible rules). We used teams of agents of different size ranging from 1 (single-agent) to 5 agents. Planning actions are associated to the agents according to the type of agent (transport, communication, assistant). Defeasible rules are distributed among agents independently of the agent type, thus making agents be able to extract context inferences about any issue of the AmI environment. This implies that the more agents, the more distributed the context information and, consequently, the fewer choices for an agent to link information and derive beliefs (arguments). As we will see later in Section 2.6.7.2, this has an impact on the obtained results.

We performed several tests varying the number of agents of each type in the AmI environment, and we took the median values over 20 repetitions for each set of experiments with *n* agents, independently of the type of agent. We used the SUM heuristic in the Plan Selection Process.

[1] A ground action is a planning action with all its parameters instantiated. Similarly with defeasible rules.

2.6.7.2 Experimental Results

In this section, we show the assessment measures we used for testing CAMAP and TMAP. Specifically, we are interested in assessing the performance and scalability of both systems, the quality and feasibility of the solution plans, and the level of trust and contribution among agents.

Performance and Scalability. For evaluating the performance and scalability, we measured the computational time of CAMAP and TMAP to find a solution plan. Figure 2.33 shows the average time spent by each system on each stage. Figures do not include the time of parsing the problem file and grounding the actions and defeasible rules. The horizontal axis depicts, for each protocol, the number of ambient agents and the size of the planning problem. The vertical axis displays the time in seconds to find a solution plan.

As expected, CAMAP always takes more time than TMAP to find a plan due to the following reasons:

(i) in the Plan Proposal Process, ambient agents in CAMAP do not only have to reason about which actions would achieve the selected open goal, but also about which arguments would support it;

(ii) since TMAP does not involve argumentation, the Plan Evaluation Process is not carried out in this protocol;

(iii) in the Plan Selection Process, TMAP simply applies the SUM heuristic to select the base plan whereas CAMAP previously executes a procedure to filter out the defeated plans from the Plan Evaluation Process; and,

(iv) in the Goal Selection Process, ambient agents in CAMAP work with a larger number of open goals, including those motivated by the argument steps.

Figure 2.33 also shows the evolution of the computational cost as the number of agents in a team increases. Obviously, the more agents in a team, the more messages exchanged between them, making each stage be much more costly. Figure 2.34 corroborates this fact. The horizontal axis of Figure 2.34 depicts the number of ambient agents, the protocol (the first three bars show CAMAP results and the other three bars represent TMAP results), as well as the size of the planning problems (green for small, red for medium, and blue for

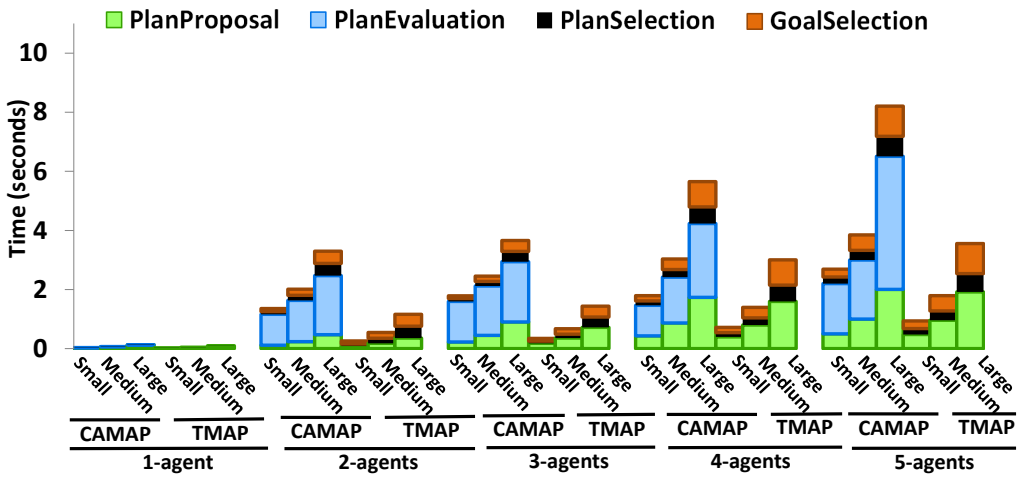

Figure 2.33: Evaluating the average time spent on each stage.

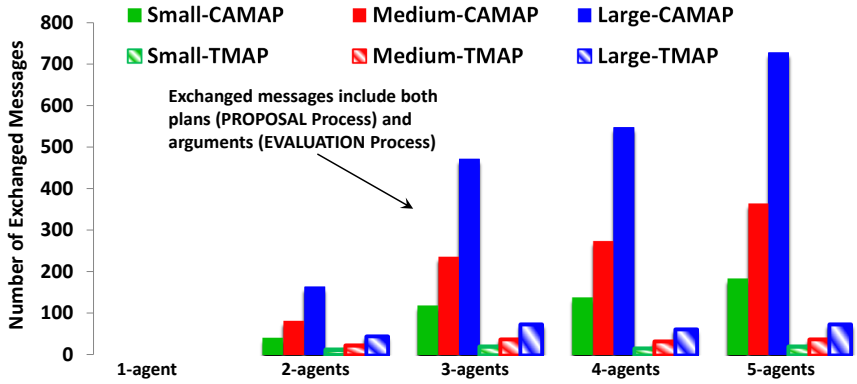

Figure 2.34: Evaluating the total number of exchanged messages between ambient agents.

large problems); the vertical axis displays the number of exchanged messages. As it can be observed, the number of exchanged messages is far larger in CAMAP than TMAP due to the exchange of arguments, which are encapsulated as messages as well.

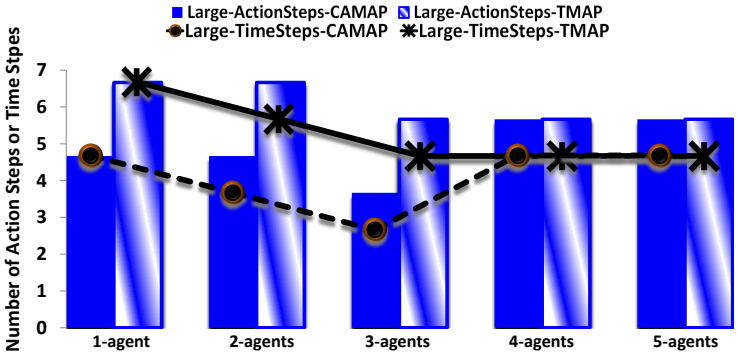

Figure 2.35: Evaluating the quality of the solution plans obtained for large difficulty: average number of action steps and average number of time steps.

Quality of the Solution Plans. In this section, we assess the quality of the solution plans according to two parameters:

- the cost of the solution plans; we assume that all action steps have one-unit cost and argument steps have no added cost.
- the number of time steps or execution steps of the solution plan; at each time step several actions can be executed in parallel by different agents.

We only show the results for large-size problems as the results are similar for small-size and medium-size problems. In order to evaluate the cost of the solution plan, Figure 2.35 shows, for each agent team and system, the average number of action steps in a solution plan. For example, for 2-agent teams, the average number of action steps of a CAMAP solution plan is 4.6 (first bar) and 6.6 for TMAP (second bar). In general, Figure 2.35 shows that the average number of action steps in solution plans of CAMAP is lower or equal than the average number in solution plans of TMAP. The reason is that in TMAP, an open goal that is not a threat, can only be achieved by an action step, while in CAMAP the open goal can also be supported by an argument step whose base is guaranteed in some state of the world generated during the planning process. In these cases, the cost of CAMAP plans is smaller because it contains fewer actions since agents' beliefs are also used to support the fulfillment of an open goal. In CAMAP, we can also observe that the average number of action steps in 4-agent and 5-agent teams is significantly higher than

for the rest of team sizes. As we explained in Section 2.6.7.1, when defeasible rules are widely distributed among agents, they are likely not to have enough information to link and build an argument step, thus increasing the number of action steps in the plan.

Regarding the comparison of time steps, plans of the of the 1-agent team are sequential plans as only one action can be executed at a time by the single agent of the team. This result contrasts with the time steps of the other teams, where actions in the plan can be executed in parallel depending on the number of agents. Obviously, the number of time steps for the 1-agent team is far more larger than for the rest of teams. On the other hand, the difference of time steps between CAMAP and TMAP is rather noticeable because it is usually the case that the fewer actions in a plan, the fewer time steps.

Feasibility of the Solution Plans. Given a context and a TMAP solution plan, we say that a plan is *feasible* if it does not contain actions that would be otherwise discarded in a CAMAP solution. That is, feasibility is a rough measure to know if a TMAP solution would contain action steps that CAMAP has labeled as failing actions as a result of the argumentative dialogues among the agents (defeated arguments) and according to the context information. Obviously, depending on the environment, feasibility can be more or less important. In our health-care scenario, feasibility is a very relevant issue. In this sense, we wanted to compare the plans returned by both systems and see how many plans, and actions correspondingly, of TMAP were actually discarded by the agents in CAMAP during the argumentative dialogues.

We carried out an experiment to count the number of actions in a TMAP solution plan that were discarded in the CAMAP counterpart. The results of this experiment are shown in Figure 2.36. As can be seen, TMAP solutions included at least 30% of action steps that CAMAP agents acknowledged not to be successfully executed. For medium and large size problems, this percentage increases considerably. However, there are hardly differences in the number of failing actions between the agent teams, an indication that feasibility is not dependent on whom proposes the actions or how many agents a team has.

Trust and Contribution. Finally, we were also interested in checking the contribution and trust level achieved by ambient agents in CAMAP. More specifically:

- The trust level for an agent is calculated as the number of undefeated argument steps in all of the plans proposed by an agent divided by the total number of argument

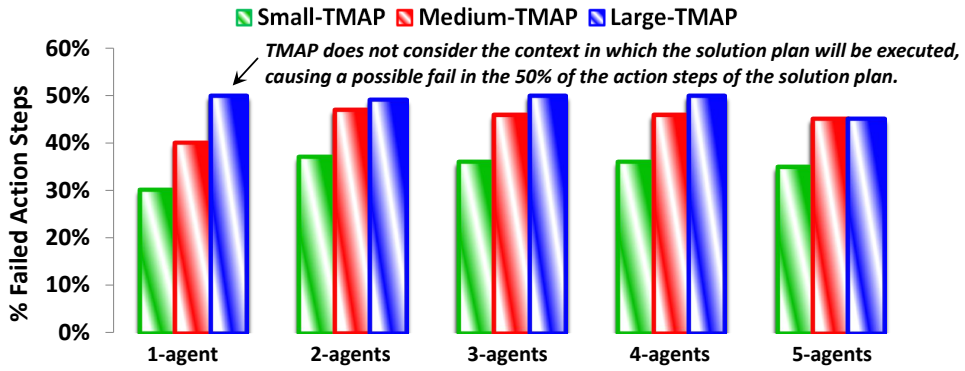

Figure 2.36: Evaluating the % of failing actions of the obtained solution plans.

steps proposed by the same agent in the Plan Proposal Process.

- The contribution level of an agent to a solution plan is calculated as the number of action and argument steps contributed by the agent to the solution plan divided by the total number of plan steps.

Our hypothesis is that agents with a high level of trust would normally have a high degree of contribution in the solution plan. Figure 2.37 shows the trust level of each agent (depicted in the vertical axis) in each experiment. We performed one experiment per team size and problem size. The results shown in Figure 2.37 indicate that the more defeasible rules known by the agents, the lower trust level achieved by the agents. For instance, in Figure 2.37 (2-agent team), agent Ag2 obtains 33% of trust level in small-size problems but 12% in large-size problems. The reason is that in large problems, agents have more defeasible rules and so they can build more context inferences about the environment, which in turn means they are likely to have more information to attack the proposal of other agents; in short, this results in fewer undefeated arguments.

Figure 2.38 shows the contribution of each agent in the solution plan returned by each team. For instance, in the 3-agent team for small problems, the individual level of contribution to the solution plan is as follows: 25% steps proposed by ambient agent Ag_1, 50% proposed by Ag_2 and 25% proposed by Ag_3. According to our hypothesis, if Ag_2 is the agent with the highest contribution in this solution plan, then Ag_2 should have a

Ag5													0%	0%	33%	
Ag4										20%	10%	0%	11%	11%	11%	
Ag3							20%	20%	20%	20%	40%	8%	0%	0%	0%	
Ag2				33%	25%	12%	40%	25%	25%	40%	10%	40%	33%	33%	16%	
Ag1	50%	40%	33%	16%	13%	0%	25%	0%	0%	25%	10%	8%	30%	30%	30%	
	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	
	CAMAP 1-agent			CAMAP 2-agents			CAMAP 3-agents			CAMAP 4-agents			5-agents			

Figure 2.37: Evaluating the agents' trust level based on their proposed argument steps.

Ag5													0%	0%	25%	
Ag4										0%	0%	0%	0%	25%	25%	
Ag3							25%	50%	25%	25%	50%	25%	0%	0%	0%	
Ag2				50%	75%	100%	50%	50%	75%	50%	25%	50%	50%	50%	25%	
Ag1	100%	100%	100%	50%	25%	0%	25%	0%	0%	25%	25%	25%	50%	25%	25%	
	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	
	CAMAP 1-agent			CAMAP 2-agents			CAMAP 3-agents			CAMAP 4-agents			5-agents			

Figure 2.38: Evaluating the agents' contribution level in the solution plans.

high individual trust level in this team. As we can see in Figure 2.37, Ag_2 has the highest individual trust level for this experiment.

2.6.7.3 Discussion

This section shows the advantages and disadvantages of CAMAP with respect to other existing approaches. Specifically, the experimental results support two main advantages of CAMAP:

(1) since each plan step of a plan proposal is collaboratively argued, CAMAP returns plans whose actions are not likely to fail at execution time according to the information and beliefs of the ambient agents; and

(2) since open goals can also be supported by the beliefs of the agents, the context information and defeasible contextual reasoning of agents provide a means to satisfy goals of the problem. This contrasts with a classical planning system where goals must always be supported by the inclusion of an action that achieves the desired effect. Therefore, CAMAP introduces a very important functionality: it allows agents to use their knowledge and context inferences to actually infer that a goal holds in the environment and that no planning action is needed to meet the goal.

It is important to highlight the relevance of aspect (1) in health-care applications where the patient' life depends, in many cases, on a timely successful execution of the plan. We can conclude that planning systems that ignore the changes in the context information are not adequate to tackle health-care applications. On the other hand, the only limitation of CAMAP in comparison to classical MAP systems is that it requires more exchange of messages between agents, which results in an increase of the computational cost.

2.6.8 Conclusions and Future Work

This article presents the specification, implementation and an exhaustive experimentation of CAMAP, a cooperative and distributed planning framework that uses defeasible argumentation to reason about the context information on smart environments. Our most relevant contribution is to come up with a fully implemented MAP framework that has been extensively tested in AmI environments, particularly on health-care applications. CAMAP realizes three independent but cooperative processes in order to *propose*, *criticize*, *defend* and *select* alternative plan proposals. All in all, the novelty in CAMAP is that of proposing a multi-agent system where ambient agents have the ability of planning while simultaneously doing context-aware reasoning. This allows agents to continuously adapt a health-care plan by performing defeasible argumentation on the beliefs of the other agents.

As future work, we would like to extend CAMAP to allow agents to put forward arguments that support or argue upon the accuracy, unambiguity and reliability of the context aware information as well as the trust level between ambient agents. Trust can be used as a preference criterion for comparing attacking arguments and resolving conflicts by prioritizing the argument with the highest trust level. Finally, we also intend to augment CAMAP to include agents' preferences and, thus, return the solution plans most preferable by the ambient agents (109, 110). For instance, an elderly patient may prefer to be hospitalized at home rather than in a hospital. This kind of preferences may be taken into account by ambient agents during the construction of the plan.

Acknowledgements

This work is mainly supported by the Spanish Ministry of Science and Education under the FPU Grant reference AP2009-1896 awarded to Sergio Pajares Ferrando, and projects,

TIN2011-27652-C03-01, and Consolider Ingenio 2010 CSD2007-00022.

2.7 Selected Paper 6: Argumentation-based Planning (Submitted AIJ 2015)

Abstract. *This paper proposes a planning system that uses defeasible argumentation to reason about context information during the planning process. The system is designed to operate in cooperative multi-agent environments where agents are endowed with planning and argumentation capabilities. Planning allows agents to contribute to the construction of the plan with actions, and argumentation is the mechanism that agents use to defend or attack the planning choices according to their beliefs. We present the formalization of all the components of the model and we provide a novel specification of the qualification problem. The multi-agent planning system, which is designed to be domain-independent, is evaluated with two planning tasks from the problem suites of the International Planning Competition. We compare our system with a non-argumentative planning framework and with a different approach of planning and argumentation. The results will show that our system is able to obtain less costly and more robust solution plans.*

2.7.1 Introduction

One common problem in Artificial Intelligence (AI) is to select the best course of action for an agent; i.e, reasoning about *what to do*. This problem has been primarily addressed from two standpoints: the knowledge or epistemological perspective, which puts the emphasis on the representation of the world such that the solution of a problem follows from the representation, and the reasoning or heuristic perspective, mostly concerned with the information for solving the problem and the reasoning process on an abstract and formal representation of the world (1). *Practical reasoning*, the research line mostly focused on the epistemological view, includes a great deal of epistemic reasoning, directed at determining what to believe (2). *Automated planning*, on the other hand, is concerned with the computational process for the selection and organization of the actions. Back in the 90's, Pollock concluded that since epistemic cognition is defeasible, a planning agent must be prepared to revise its plans as its defeasibly held beliefs change and may have to acquire more information through reasoning to solve a planning problem (3).

The mainstream in practical reasoning lies in the use of argumentation theory so as to extend the means-end reasoning in classical planning with presumptive justifications for the adoption of a particular action. Using Dung's argumentation framework over beliefs

(4) has been the predominant approach in practical reasoning, like a proposal for arguing about what desires an agent should adopt and about what plans to intend in order to achieve these desires (5); the study of the goal deliberation process (6); or the generation of consistent plans from a set of conflicting beliefs (7). Building argumentation plans for negotiating conflict resolution at a planning stage is also an interesting application of argumentation in practical reasoning (8). Some other works, however, follow the notion of argument scheme proposed by Walton (9) and present an approach in which arguments and conflicts are represented as argument schemes and critical questions, respectively (10). This latter work has been one of the most popular approaches in practical reasoning, it has demonstrated its applicability in domains such as law, experimental economics or e-democracy (13, 14, 15) and it has also been exploited for the design of argumentation-based dialogues to support automated coordination in distributed planning (16), multi-agent deliberation dialogues (17) or the construction of joint plans (18).

Unlike argumentation-based approaches of practical reasoning, another line of investigation closer to *planning* also explores the relationships between classical planning and argumentation but building upon a planning formalism and using argumentation to guide the reasoning process. A first step in this direction assumes that agent's deductions are not always certain information, but *plausible*, and the conclusions can be withdrawn when new pieces of knowledge are found; i.e., agents must use defeasible reasoning (19). OSCAR is a goal-regression planner that essentially performs the same search of Partial-Order Planning (POP) but reasoning defeasibly about candidate plans at the end of the planning process (20). In OSCAR, the plan search itself is also done defeasibly, thus enabling to reason about the impact of unexpected environmental conditions on the solution plans and to select the plan which is less likely to fail at execution time. In the same line, another pioneer work presents a formal model of plans based on a defeasible argument system that is able to suggest aspects of a plan, criticize a plan and revise the plan (21). Both of these investigations, considered as the first steps towards building an *argumentation-based planning* system, have clear similarities to the works on plan modification and replanning but rather than forcing the planner to resort to replanning in light of new information, they consider planning within the context of a general defeasible reasoning system.

More recently, Simari et al present a defeasible argumentation framework for the definition of actions and the combination of these actions into plans (22). This work lays

the foundations of an argumentation-based formalism for constructing plans (23) by using Defeasible Logic Programming (DeLP) (24), a formalism that agents use to represent their knowledge and build applications that deal with incomplete and contradictory information in dynamic domains. The formalism presented in (23), which we will refer to as DeLP-POP in the following, describes how the traditional POP algorithm can be extended to consider arguments as planning steps.

Subsequently, further investigations on argumentation-based planning focus on the application of argument-based systems to Multi-Agent Planning (MAP). An argumentation-based dialogue protocol that enable agents to discuss candidate plans and reach agreements was proposed in (29, 30), where candidate plans of the agents are generated by an external single-agent planner and the protocol is used then for reasoning about the contradictory planning beliefs in the candidate plans and select a valid solution plan. In this approach, agents use argumentation to defend or attack the candidate plans put forward by others, but not for cooperatively building a plan contributed by several agents. Another interesting work that combines the benefits of using argumentation in MAP emphasizes the utilization of argumentation to solve conflicts between sub-plans of different agents by means of deliberative dialogues based on argumentation schemes (31, 32). Conflicts may be caused by concurrent actions, plan constraints or norms the agents must adhere to and argumentation is used as a mechanism for analyzing these conflicts when several sub-plans of different agents have to be merged. Likewise, argumentation is not used for building plans but for arguing at end of the planning generation, a task that is accomplished by an external classical planner. A different approach that also makes use of argumentation schemes proposes structured argumentative dialogues to coordinate plan-related tasks (16). Particularly, agents coordinate their beliefs and intentions with the use of a strategy based on an argumentation scheme and a set of related critical questions for selecting plan proposals. Thus, choosing an appropriate question in the dialogue becomes an important issue in terms of dialogue and cooperation efficiency.

On the other hand, the work in (26) presents a formal extension of DeLP-POP to a multi-agent context in which agents are assumed to have planning and argumentation capabilities. Specifically, this work proposes a formal dialogue for an incremental argumentative plan search, by which agents exchange plan proposals and arguments for or against such proposals. To the best of our knowledge, this work represents the first attempt to use an argumentation-based MAP mechanism for a cooperative construction of

plans. Subsequently, the works in (27, 28, 34) present a first practical extension for evaluating the approach in (26) in a transit journey planning service and in ambient intelligent applications.

In this paper, we present the formalization of Q-DeLP-POP and its extension to a MAP environment (Q-DeLP-MAP), an argumentation-based MAP system that elaborates on two previous contributions: an initial formalization of a multiagent argumentative planning model in the framework of DeLP-POP (26), and a preliminary implementation of such argumentative MAP model in a domain of ambient intelligent applications (27, 28). The results obtained in these latter works revealed that the argumentation-based MAP model was able to deal with rich argumentative representations but had a limited planning capability. Q-DeLP-MAP, however, greatly outperforms the previous system by exploiting, among other things, the reuse of argumentative dialogues during the construction of a POP search tree, which allows us to tackle problems from the International Planning Competitions (IPC)[1]. Additionally, Q-DeLP-POP provides a more sophisticated specification of the *qualification problem* in planning, defining novel relationships between argument steps and action steps of a plan. Overall, the aim of this paper is to put together and exploit the investigations carried out in (26) and (28) in order to come up with a domain-independent, fully integrated and operative argumentation-based MAP model.

This paper is organized as follows. Section 2.7.2 summarizes the main foundations in which this work is based on. Section 2.7.3 presents in detail the components of our defeasible-argumentation-based planning framework. Section 2.7.4 presents the multiagent protocol for our planning framework. Next, the experiments carried out to validate the present work are described and analyzed. Finally, we conclude and present some directions for future work.

2.7.2 Preliminary notions

In this section, we summarize the foundations used throughout this paper in order to facilitate the understanding of our proposal that will be presented in the following sections.

[1] http://ipc.icaps-conference.org/

2.7.2.1 DeLP: a framework for defeasible argumentation

Defeasible Logic Programming (DeLP) is a framework for reasoning about defeasible information in single-agent contexts (24). Defeasible reasoning is a process where tentative conclusions are obtained from uncertain or incomplete information, and conclusions might no longer be valid after new information becomes available (111). DeLP is one popular approach to make context inferences by using defeasible argumentation, a particular type of defeasible reasoning.

The key element of DeLP is the *defeasible rules* (Head $\prec$ Body), which are used to represent a deductive relation between pieces of knowledge that could be defeated once another piece of knowledge is considered.

In (24), the authors propose a non-monotonic consequence relation, called *warrant*, built upon the relation of defeat between constructible arguments for or against a literal. The agent's knowledge base is a pair $T = (\Psi, \Delta)$ consisting of a strict and a defeasible part:

- a consistent set $\Psi \subseteq$ Lit of *facts*, and
- a set Δ of defeasible *rules* $\delta = \ell \multimap \ell_0, \ldots, \ell_k$

where $\ell, \ell_0, \ldots, \ell_k \subseteq$ Lit. Rule $\ell \multimap \ell_0, \ldots, \ell_k$ expresses that a warrant for $\ell_0, \ldots, \ell_n$ provides a (defeasible) reason for ℓ to be warranted[1]. We denote $\mathsf{body}(\delta) = \{\ell_0, \ldots, \ell_n\}$ and $\mathsf{head}(\delta) = \ell$ as, respectively, the *body* and *head* of δ.

Given T, a derivation for a literal ℓ from T is called *'defeasible'* because there may exist information in contradiction with ℓ that will prevent the acceptance of ℓ as a valid conclusion. For example, let's consider:

- $\Psi = \{connection(space, earth),\ solar\_storm(space)\}$, and
- $\Delta = \{\delta_0 = communicated\_rock\_data(space, earth) \prec connection(space, earth),$
  $\delta_1 = \sim communicated\_rock\_data(space, earth) \prec solar\_storm(space)\}$

where the first fact of Ψ means that there is a connection between the space and the earth, and the second that it exists a solar storm at space. δ_0 denotes that there are reasons to believe the rock data are successfully communicated from the space to the

[1] Strict rules introduced in (66) (24) have not been considered in planning, see (23).

earth. However, δ_1 provides reasons to believe the contrary, in whose case we say that the piece of information derived by δ_0 is acknowledged to fail[1]. Thus, the set of derivable literals in T will be not in general consistent. Additionally, there could be more one defeasible derivation for a literal ℓ (rule set consisting of one or more defeasible rules) and more than one defeasible derivation against.

Defeasible argumentation is a form of defeasible reasoning that emphasizes the notion of an argument. An argument is a chain of reasoning that concludes one piece of information (conclusion) on the basis of some other pieces of information (premises). An argument $\mathcal{A}$ for a literal ℓ in T is a subset of defeasible rules $\mathsf{Rul}(\mathcal{A}) \subseteq \Delta$, denoted as $\langle \mathsf{Rul}(\mathcal{A}), \ell \rangle$ or simply $\mathcal{A}$, such that:

(i) ℓ is derivable from $\Psi \cup \mathsf{Rul}(\mathcal{A})$,

(ii) the set $\Psi \cup \mathsf{Rul}(\mathcal{A})$ is non-contradictory, and,

(iii) $\mathsf{Rul}(\mathcal{A})$ is a minimal subset of Δ satisfying (i) and (ii).

We also define, for an argument $\mathcal{A}$ for ℓ (26):

$$\begin{aligned} \mathsf{base}(\mathcal{A}) &= \mathsf{body}(\mathsf{Rul}(\mathcal{A})) \setminus \mathsf{head}(\mathsf{Rul}(\mathcal{A})) \\ \mathsf{concl}(\mathcal{A}) &= \mathsf{head}(\mathsf{Rul}(\mathcal{A})) \setminus \mathsf{body}(\mathsf{Rul}(\mathcal{A})) \\ \mathsf{literals}(\mathcal{A}) &= \mathsf{body}(\mathsf{Rul}(\mathcal{A})) \cup \mathsf{head}(\mathsf{Rul}(\mathcal{A})) \end{aligned}$$

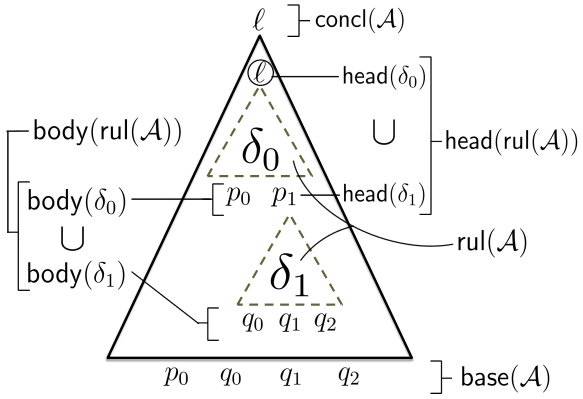

Figure 2.39: An argument $\mathcal{A}$ for l composed of two rules δ_0, $\delta 1 \in \mathsf{Rul}(\mathcal{A})$.

[1]Notation: we use both $\sim p$ and $\overline{p}$ to deny the literal p, s.t. $\overline{p} = \sim p$.

The argument $\mathcal{A}$ shown in Figure 2.39 is encoded as follows: $\Psi = \{p_0, q_0, q_1, q_2\}$, and $\Delta = \{\delta_0 = \ell \prec \{p_0, p_1\}, \delta_1 = p_1 \prec \{q_0, q_1, q_2\}\}$. The existence of $\mathcal{A}$ for a literal ℓ, i.e. $\mathsf{concl}(\mathcal{A}) = \ell$, still, does not suffice for ℓ being warranted; we must guarantee that $\mathcal{A}$ is not defeated by some other argument. More specifically, given two arguments $\mathcal{A}$ and $\mathcal{A}_1$, where $\{\mathsf{base}(\mathcal{A}) \cup \mathsf{base}(\mathcal{A}_1)\} \subseteq \Psi$, we say that $\mathcal{A}_1$ attacks $\mathcal{A}$ if the conclusion of $\mathcal{A}_1$ contradicts some literal derived in $\mathcal{A}$, that is, if $\overline{\mathsf{concl}(\mathcal{A}_1)} \in \mathsf{head}(\mathsf{Rul}(\mathcal{A}))$. The attack relation may roughly be seen as symmetric, in the sense that each attacked argument $\mathcal{A}$ contains a sub-argument $\mathcal{A}'$ attacking $\mathcal{A}_1$. To decide which contending argument prevails in an attack, a notion for preference among pairs of conflicting arguments is needed. In the rest of this article, we follow a formal criterion based on a comparison of information used by each argument: an argument which makes use of more precise rules (or more information) is a *proper defeater* for -is preferred to- the contending argument (24). If two contending arguments are not comparable in these terms, they are a *blocking defeater* for each other[1]. As shown in Figure 2.40(a), a counter-argument $\mathcal{A}_1$ can attack directly the conclusion of an argument $\mathcal{A}$, or as illustrated in Figure 2.40(b), it can attack an inner point of $\mathcal{A}$.

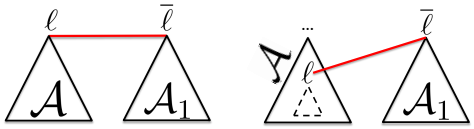

Figure 2.40: Attack types: (a) a direct attack, and (b) an indirect attack.

Given an argument $\mathcal{A}_0$ for ℓ, an *argumentation line* $\Lambda = [\mathcal{A}_0, \ldots, \mathcal{A}_n]$ is a sequence of arguments constructible from (Ψ, Δ), where each argument $\mathcal{A}_{k+1}$ is a defeater for its predecessor $\mathcal{A}_k$. Arguments supporting (resp. interfering with) $\mathcal{A}_0$, i.e. of the form $\mathcal{A}_{2n}$ (resp. $\mathcal{A}_{2n+1}$) must form a consistent set, and no sub-argument $\mathcal{A}'$ of an argument $\mathcal{A}_m \in \Lambda$ may appear later in Λ (i.e. it cannot be that $\mathcal{A}' = \mathcal{A}_{m'}$ with $m' > m$).

Given a root argument $\mathcal{A}$, the union of its argumentation lines gives rise to a tree-like structure, the *dialectical tree* for $\mathcal{A}$, denoted $\mathcal{T}_{\mathcal{A}}(\Psi, \Delta)$. Figure 2.41 shows an example of two dialectical trees.

[1] Alternatively, one can specify by hand a preference between rules and then induce a defeat relation for arguments out of it. See (66) for details.

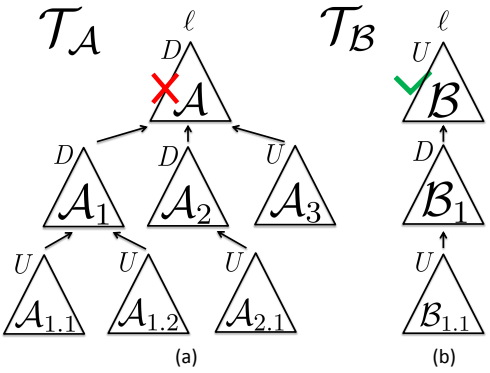

Figure 2.41: Computing warrancy for l: (a) $\mathcal{T}_{\mathcal{A}}$: $\mathcal{A}$ is a defeated argument, and (b) $\mathcal{T}_{\mathcal{B}}$: $\mathcal{B}$ is an undefeated argument.

As shown in (24), to check whether the argument of the root node of Figure 2.41 (a) and (b) is defeated or undefeated, the following procedure on the Dialectical Tree is applied: label with a *U* (for *undefeated*) each terminal node in the tree (i.e. each argument with no defeaters at all). Then, in a bottom-up fashion, we label a node with:

$$\begin{cases} U & \text{if each of its successors is labeled with a } D \\ D & \text{(for } \textit{defeated}\text{) otherwise} \end{cases}$$

The application of this procedure in Figure 2.41 (a) returns that $\mathcal{A}$ is a defeated argument, and Figure 2.41 (b) that $\mathcal{B}$ is an undefeated argument.

2.7.2.2 POP: Partial Order Planning

Partial-Order Planning (POP) is a suitable planning approach to address the requirements derived from a distributed planning thanks to the application of the least commitment principle (43, 46), which delays commitment of action orderings until a decision is necessary to solve some inconsistency. In POP, a plan is represented as a set of actions and a set of ordering constraints defining a partial order between actions. In this sense, the partial order paradigm is a flexible mechanism to deal with the individual plans of different agents and combine them into a single joint plan. Since our goal is the integration of planning and defeasible argumentation in a multi-agent context, we adopt POP as the planning approach of the agents. In this section, we show the basic foundations of POP.

A planning task is defined as a tuple $\mathbb{M} = (\Psi, A, G)$, where $\Psi \subseteq \mathsf{Lit}$ is the set of facts that represent the initial state of the planning task, A is the set of actions and, $G \subseteq \mathsf{Lit}$ is the set of goals of the planning task. An action $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$ is a set of preconditions (for α to be applicable) and effects.

A POP plan Π, or simply a plan Π, is a 3-tuple $\langle A(\Pi), \mathcal{CL}(\Pi), \mathcal{OC}(\Pi) \rangle$, where $A(\Pi)$ is the set of action steps in Π, $\mathcal{CL}(\Pi)$ is the set of causal links in Π, and $\mathcal{OC}(\Pi)$ is the set of ordering constraints on $A(\Pi)$. A *causal link* between two actions: α_i and α_j, is denoted as $(\alpha_i, \ell, \alpha_j) \in \mathcal{CL}(\Pi)$, meaning that $\ell \in \mathsf{P}(\alpha_j)$ is planned to be supported by α_i. In addition, for a certain pair of action steps α_i and α_j, α_i may precede α_j or viceversa. Such a relationship is called *ordering constraint* and is denoted as $\alpha_i \prec_\Pi \alpha_j$ (or $\alpha_j \prec_\Pi \alpha_i$) $\in \mathcal{OC}(\Pi)$. It is important to note that not every pair of actions has to be ordered: for a given pair of action steps, it may be that neither action step precedes the other in the plan. In POP, Ψ and G are encoded as dummy actions $\alpha_\Psi \prec_\Pi \alpha_G$ with $\mathsf{X}(\alpha_\Psi) = \Psi$, $\mathsf{P}(\alpha_G) = G$ and $\mathsf{P}(\alpha_\Psi) = \mathsf{X}(\alpha_G) = \emptyset$. Finally, the set of unsupported preconditions of the action steps in Π is called *open goals*, and is denoted as $\mathsf{goals}(\Pi)$.

$A(\Pi)$ and $\mathcal{CL}(\Pi)$ in a plan may cause the appearance of *threats* in POP: an action β potentially interfering with a causal link $(\alpha_i, \ell, \alpha_j)$ is denoted by $(\beta, (\alpha_i, \ell, \alpha_j))$. A threat means that the interfering action would invalidate the causal link if this step is ordered between the two actions of the causal link. The set of all threats in a plan Π is labeled as $\mathsf{threats}(\Pi)$, and initially, $\mathsf{threats}(\Pi) = 0$. When detected, threats are to be solved by some threat resolution step: demote or promote.

The set of flaws to be solved in a plan Π includes $\mathsf{threats}(\Pi)$ and $\mathsf{goals}(\Pi)$, and is denoted as $\mathsf{flaws}(\Pi)$. A plan Π is solution if $\mathsf{flaws}(\Pi) = \emptyset$; i.e, if Π is a threat-free plan and all the goals in G are achieved through a causal link. If Π is a solution plan, then $A(\Pi)$ applied over the initial state Ψ leads to a problem state in which the goals of the task, G, hold.

2.7.2.3 DeLP-POP: A first extension of POP with DeLP

DeLP-POP is a theoretical extension of POP with DeLP-style argumentation (23). A DeLP-POP planner can appeal both to arguments and actions as a way to resolve goals or threats. DeLP-POP distinguishes between two types of steps for supporting goals: actions steps and argument steps. Actions are intended to express the physics of a domain so the effects of an action reflect the changes that will be produced in the world when the

action is executed. However, argument steps represent the conclusion inferred by an agent according to its local knowledge and partial view of the world. The novelty of DeLP-POP is that arguments can be introduced in the plan to support action preconditions. Consequently, the conclusion derived through an argument may be invalidated if another agent puts forward an opposite conclusion.

A planning task $\mathbb{M}$ is extended in DeLP-POP as a tuple (Ψ, Δ, A, G), where $\Psi \subseteq \mathsf{Lit}$ is the set of initial facts, the new element Δ contains defeasible rules that may apply in any of the world states that result from the execution of the plan, A is the set of actions, and $G \subseteq \mathsf{Lit}$ is the set of goals. The new element Δ allows the agent to construct arguments during the planning search.

Let ℓ be an open goal, motivated by some step $\beta \in A(\Pi)$ or $\mathcal{A} \subseteq \Delta$; i.e. $\ell \in \mathsf{P}(\beta)$ or $\ell \in \mathsf{base}(\mathcal{A})$. If goal ℓ is planned to be enforced by an action α, this is encoded as a *causal link* of Π and included in the set $\mathcal{CL}(\Pi)$: $(\alpha, \ell, \kappa) \in \mathcal{CL}(\Pi)$, with $\kappa = \beta$ or $\kappa = \mathcal{A}$. If goal $\ell \in \mathsf{P}(\beta)$ is to be enforced by an argument, this is encoded as a *support link* of Π, in a set denoted $\mathcal{SL}(\Pi)$: $(\mathcal{B}, \ell, \beta) \in \mathcal{SL}(\Pi)$, where $\mathcal{B} \subseteq \Delta$.

A plan Π for a DeLP-POP task $\mathbb{M}$, is a 5-tuple $\langle A(\Pi), \mathcal{AR}(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi), \mathcal{OC}(\Pi)\rangle$, where $A(\Pi)$ denotes the set of action steps; $\mathcal{AR}(\Pi)$ represents the set of argument steps, or more particularly, the utilization of the defeasible rules in Δ within Π; $\mathcal{CL}(\Pi)$ and $\mathcal{SL}(\Pi)$ represent the sets of causal and support links, respectively; and $\mathcal{OC}(\Pi)$ is a set of ordering constraints.

In DeLP-POP, arguments are not only introduced to intentionally support an action precondition of a plan, but they are also presented to defeat or defend other arguments in the plan.

When actions and arguments are combined in a partial order plan, new types of threats appear and need to be solved (23, 112). In section 2.7.3.2, we provide some more details on the DeLP-POP threats and we compare them with the definition of threats in our proposal.

2.7.3 Components of Q-DeLP-POP

In this section, we introduce the semantics of a new planning framework, Q-DeLP-POP, that extends DeLP-POP (23) for dealing with the *qualification problem* (59). As we will see, the adopted solution to the *qualification problem* leads to redefine some of the components of the DeLP-POP framework (23) as well as to introduce new ones. Q-DeLP-POP

has been previously applied to scenarios that range from applications of ambient intelligence in the field of health-care (27, 28) to a transit journey planning service (34, 73). In this work, we present a detailed and thorough formalization of Q-DeLP-POP.

The motivation behind Q-DeLP-POP arises from the fact that in natural environments the successful execution of actions can never be predicted with absolute certainty. Unexpected circumstances, albeit unlikely, may at any time prevent an autonomous agent from performing the intended actions (113).

Example 1. *An air traffic control strike is due to take place in Munich. An action that involves taking a plane at the Munich airport could fail during the plan execution, thus invoking a plan repair or replanning procedure at execution time. If the announcement of the strike were known during the construction of the plan, an alternative journey route with no stops in Munich would have been selected.*

The information about the controllers' strike at the Munich airport of Example 1 is one of the many unexpected conditions that might prevent the user from executing an action such as taking a plane in Munich. The qualification problem is concerned with the impossibility of listing all the preconditions required for a real world action to have its intended effect.

Within the planning community, the default choice to the qualification problem is to assume away the numerous possible unexpected circumstances that may prevent an action from being executed in the real world and resort to replanning in case a plan turns out not to be executable due to a non-anticipating condition. Some approaches suggest treating actions as qualified by general constraints describing the domain being investigated such that the action is computed without considering these domain constraints, and the action is only taken to qualify if and only if any of the constraints is violated after the computation is complete (59). Other method though cope with the qualification problem by respecting causality when minimizing anomalous models, which requires a prior definition of abnormal qualifications and anomalous models (113). Our method to overcome the qualification problem during the construction of a plan relies on the local knowledge of the agent about anomalous situations. Rather than verifying all unusual situations in the preconditions of the actions, agents put forward an action to the plan in the form of an argument so that any one agent that has information about a tentative anomalous situation that might prevent the action from being executed launches an attack against the argument representing the action. This new semantic representation will be shown hereinafter.

2.7.3.1 Action-Argument Steps

Q-DeLP-POP provides a novel representation mechanism for action specifications, through which agents are able to attack an action step of the plan if they hold information about an environmental condition that could potentially prevent the action from having its intended effects. Given an action α that is to be inserted in a plan Π as an action step of $A(\Pi)$, the underlying idea is to represent the generation of $\mathsf{X}(\alpha)$ through a defeasible derivation and encode such a derivation as an argument of Π. For this purpose, Q-DeLP-POP replaces the notion of action step (elements in $A(\Pi)$) by a new compound entity called **action-argument step**, whereas it keeps the notion of argument step ($\mathcal{AR}(\Pi)$) as defined in DeLP-POP.

Definition 6. [Action-Argument Step]. *Let α be an action with $\mathsf{X}(\alpha) = \ell$ to be inserted in a plan Π. In* Q-DeLP-POP*, α is inserted in Π as a pair action-argument $\gamma = \langle \alpha', \mathcal{A} \rangle$. Particularly:*

- $\mathsf{X}(\alpha') = \mu_{\alpha'}$*, where $\mu_{\alpha'}$ is a ficticious and irrevocable effect used to denote the actual execution of action α'.*
- $\mathsf{Rul}(\mathcal{A}) = \ell \multimap \mu_{\alpha'}$ *where* $\mathsf{base}(\mathcal{A}) = \mu_{\alpha'}$ *and* $\mathsf{concl}(\mathcal{A}) = \mathsf{X}(\alpha)$.
- $\mathsf{P}(\gamma) = \mathsf{P}(\alpha') = \mathsf{P}(\alpha)$.
- $\mathsf{X}(\gamma) = \mathsf{concl}(A) = \mathsf{X}(\alpha)$.

Thus, in Q-DeLP-POP, action steps are all replaced by action-argument steps. This leads to a redefinition of a plan Π as a 5-tuple $\langle AA(\Pi), AR(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi), \mathcal{OC}(\Pi) \rangle$, where $AA(\Pi)$ represents the set of action-arguments in Π. We will call the set $\mathcal{AR}(\Pi)$ *supporting arguments* so as to distinguish them from the arguments comprised in $AA(\Pi)$, which are simply a type of fictitious arguments artificially created to transform the effect of an action into a defeasible derivation. Identifying $\mathcal{AR}(\Pi)$ as the set of supporting arguments of the plan Π will also be helpful to characterize this set of arguments versus the *attacking arguments*, a distinct type of arguments that will be later introduced. In any case, an argument always denotes a defeasible piece of knowledge or belief of the agent.

The introduction of the compound entity action-argument step also implies to revisit the notions of *causal link* and *support link*. Specifically, let $\gamma_2 = \langle \alpha'_2, \mathcal{B} \rangle$ be the action-argument of an action α_2 which is represented on the right of Figure 2.42[1]; and

[1]Note that we represent the effects of α'_1 above the rectangle of action α'_1 and its preconditions below.

let $\mathsf{P}(\gamma_2) = \mathsf{P}(\alpha'_2) = \ell$ be an open goal of γ_2; then, if ℓ is supported by the action-argument step $\gamma_1 = \langle \alpha'_1, \mathcal{A} \rangle$ of an action α_1, represented on the left of Figure 2.42, then Q-DeLP-POP introduces a *causal link* $(\gamma_1, \ell, \gamma_2) \in \mathcal{CL}(\Pi)$, as shown in Figure 2.42.

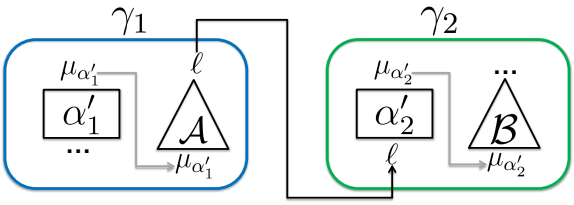

Figure 2.42: Example of causal link $(\gamma_1, \ell, \gamma_2) \in \mathcal{CL}(\Pi)$.

Similarly to DeLP-POP, where a precondition of an action step can be supported by an argument step, we allow preconditions of action-argument steps to be likewise supported by supporting arguments. Specifically, let's consider the action α_2 in Figure 2.43 represented through the action-argument γ_2. If $\ell \in \mathsf{P}(\gamma_2)$ is to be supported by an argument step $\mathcal{C}$, then Q-DeLP-POP introduces a support link $(\mathcal{C}, \ell, \gamma_2) \in \mathcal{SL}(\Pi)$ and inserts $\mathcal{C}$ in the set $\mathcal{AR}(\Pi)$. This is graphically represented in Figure 2.43 where the supporting argument $\mathcal{C}$ is used to support $\mathsf{P}(\gamma_2)$.

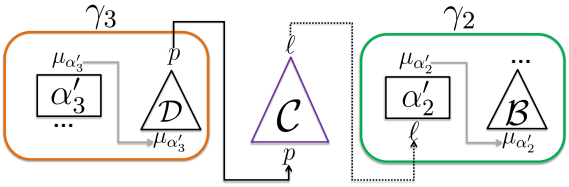

Figure 2.43: Example of support link $(\mathcal{C}, \ell, \gamma_2) \in \mathcal{SL}(\Pi)$ and causal link $(\gamma_3, p, \mathcal{C}) \in \mathcal{CL}(\Pi)$.

Finally, as it also occurs in DeLP-POP, a literal p that belongs to the base of an argument, say $p \in \mathsf{base}(\mathcal{C})$, can be supported by an action-argument $\gamma_3 = \langle \alpha'_3, \mathcal{D} \rangle$ in Q-DeLP-POP, thus introducing a *causal link* $(\gamma_3, p, \mathcal{C}) \in \mathcal{CL}(\Pi)$, as shown on the left of Figure 2.43.

In case that an action α has more than one effect, i.e., $\mathsf{X}(\alpha) = \{\ell_0, \ell_1, \ldots, \ell_n\}$, the corresponding action-argument step is denoted as a 2-tuple $\gamma = \langle \alpha, \{\mathcal{A}_0, \mathcal{A}_1, \ldots\} \rangle$, where

$\mathsf{Rul}(\mathcal{A}_0) = \ell_0 \multimap\!\prec \mu_\alpha$, $\mathsf{Rul}(\mathcal{A}_1) = \ell_1 \multimap\!\prec \mu_\alpha$, and so on. Thus, an action-argument step is a compound entity that comprises a grounded action and a set of arguments that encapsulate the effects of this action. As we will see in section 2.7.3.2, representing the effects of an action through a defeasible derivation instead of a strict derivation as it occurs in DeLP-POP, allow agents to put forward arguments for or against the successful execution of the action and the achievement of its intended effects. Note that our aim is to ensure that when the solution plan is to be executed in the real world, things happen as featured in the plan; that is, as envisioned by the agents according to the contextual knowledge defined by their beliefs.

We say that a literal $\ell \in \mathsf{Lit}$ is an open goal in Π, denoted as $\ell \in \mathsf{goals}(\Pi)$, if $\exists \gamma \in AA(\Pi) \mid \ell \in \mathsf{P}(\gamma)$ or $\exists \mathcal{A} \in AR(\Pi) \mid \ell \in \mathsf{base}(\mathcal{A})$, and it does not exist a causal link or support link for the precondition of the action-argument or a causal link for the base of the argument in Π that supports ℓ.

2.7.3.2 Conflicting situations

During the construction of a plan, different conflicting situations or interferences may arise that need to be solved in order to guarantee the validity or correctness of the plan and its feasibility or executability in the real world. In POP, interferences between actions are captured through the notion of threat, as shown in section 2.7.2.2. However, when actions and arguments are involved in the construction of a plan, new types of conflicting situations appear that need to be identified and solved in order to obtain a valid plan. Particularly, DeLP-POP extends the notion of POP threat to capture the interactions of actions with arguments and arguments with arguments. Unlike DeLP-POP (23, 112), we distinguish between two types of conflicting situations in a plan:

- *Threats*: they occur between two unordered steps of the plan such that if one is ordered before the other, the first one invalidates the application of the second one.
- *Attacks*: they occur when an agent holds some belief that may contradict the conclusions of a step of the plan. Attacks arise because of the existence of contradictory local information of the agents.

The difference between threats and attacks is as follows: while threats are used to validate the plan according to the physics expressed in the domain of the problem, attacks respond to the local and contextual information held by the agents (beliefs), which is not

expressed in the physics of the domain. Thereby, threats are aimed at checking the validity of the plan while attacks are aimed at checking the executability of the actions in the plan and the achievement of their intended effects. In what follows an example for the ease of understanding.

Example 2. *In Figure 2.44(a), the action-argument step γ_1 represents the planning action 'fly plane apn1 from Munich to London'. The argument step $\mathcal{A}$ derives 'apn1 at Munich' according to information provided by the air traffic control tower of Munich airport. The precondition of γ_1 is thus supported by the conclusion of $\mathcal{A}$. In this case, the conflicting situation occurs when a new action-argument, step γ_2, contradicts the support link $(\mathcal{A}$, apn1-at-Munich, $\gamma_1) \in \mathcal{SL}(\Pi)$. This conflicting situation is similar to the classical notion of threat in* POP *but in this case γ_2 threatens a support link. On the other hand, Figure 2.44(b) shows two arguments: argument $\mathcal{B}_1$ denotes there are reasons to believe 'a volcanic and ash cloud' might happen and argument $\mathcal{B}_2$ denotes that the agent believes 'an airport strike might occur'. $\mathcal{B}_1$ and $\mathcal{B}_2$ are not part of the plan since these arguments are not supporting any open goal of the plan, but they are a consequence of the beliefs of the agents, which may prevent γ_1 from achieving its intended effects. In other words, agents are saying that there are reasons to believe that these two unexpected circumstances (a volcanic ash cloud and airport strike) may occur and prevent the plane from flying from Munich to London. This conflicting situation is known in* Q-DeLP-POP *as an attack.*

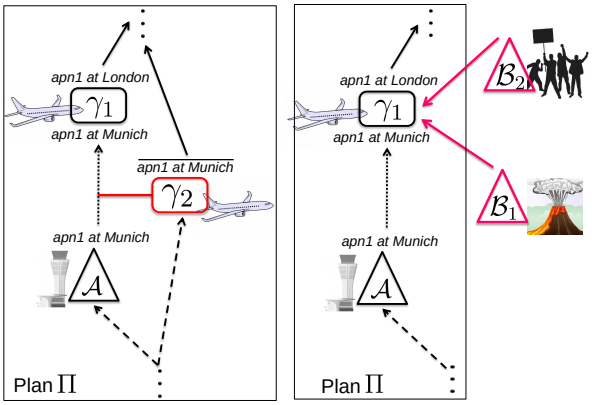

Figure 2.44: Example 2: (a) a threat in the plan; and, (b) attacks to the plan.

The following sections 2.7.3.2 and 2.7.3.2 address in detail the management of threats and attacks, respectively.

Threats. A threat happens when some step of the plan threatens the support provided by another step of the plan. Formally, we define a threat as a tuple $\langle k_3, (k_1, \ell, k_2)\rangle$, where k_1, k_2 and k_3 are action-argument steps or supporting arguments; k_3 is the threatening step which threatens the support (k_1, ℓ, k_2), being k_1 the step which provided some support to k_2. All type of threats in Q-DeLP-POP will follow this uniform definition. In DeLP-POP, however, threats are defined as a step threatening a link or a step threatening another step depending on the type of the threat (23, 112). By keeping a uniform threat format, a neater classification of threats as well as a simpler definition of the solutions can be defined. In the following, we will also point out some differences with respect to the DeLP-POP threat classification.

Threats in Q-DeLP-POP are classified according to the type of support is being threatened: a causal link, a support link or an internal support of an argument:

(1) Causal Links Threat (CLT): this interference occurs when a plan step threatens a causal link of the plan.

(2) Support Link Threats (SLT): this interference occurs when a plan step threatens a support link of the plan.

(3) Internal Support Threats (IST): this interference occurs when a plan step threatens an internal literal derived by some argument step of the plan.

Given a plan Π, and following the above threat definition $\langle k_3, (k_1, \ell, k_2)\rangle$, we define the following types of threats:

Causal Link Threat (CLT). Let (k_1, ℓ, k_2) be a causal link; then, k_1 must be an element in $AA(\Pi)$ and k_2 can be an action-argument or a supporting argument of the plan. In this case, we say k_3 is threatening a causal link of the plan. The two situations that may arise are graphically depicted in Figures 2.45(a) and 2.45(b). Figure 2.45(a) represents the classical POP threat. In Figure 2.45(b), the supporting argument $\mathcal{A} \in AR(\Pi)$ has been introduced to support some precondition of γ_2 and then γ_1 is used to support the literal ℓ of $\mathsf{base}(\mathcal{A})$. Therefore, $k_3 = \gamma_3$ is threatening the causal link $(\gamma_1, \ell, \mathcal{A})$. In both cases, 2.45(a) and 2.45(b), the threat is solved by applying demotion $(\gamma_2 \prec_\Pi \gamma_3)$

or promotion ($\gamma_1 \prec_\Pi \gamma_3$). Our CLTs are denoted in DeLP-POP as *action-action* and *action-base* threats.

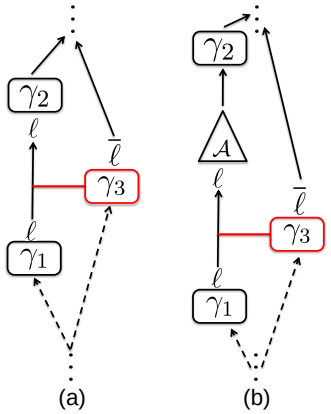

Figure 2.45: Examples of threats of type CLT.

Support Link Threats (SLT). (k_1, ℓ, k_2) is a support link if $k_1 \in AR(\Pi)$ and $k_2 \in AA(\Pi)$, and k_3, the threatening step, can be an action-argument or a supporting argument. The former case is shown in Figure 2.46(a), where $k_1 = \mathcal{B}$ and $k_2 = \gamma_2$. This threat is solved by applying demotion ($\gamma_2 \prec_\Pi \gamma_3$). The case in which $k_3 \in AR(\Pi)$ is shown in Figure 2.46(b). The threatening step is $k_3 = \mathcal{C}$, which has been introduced in the plan to support the precondition of some other action (this is not graphically represented). Although the appearance of $\mathcal{C}$ in the plan can be interpreted as an interference with other steps of Π, this type of threat is not solved until $\mathsf{base}(\mathcal{C})$ is supported by an action-argument since ordering constraints are only established between action-arguments. This does not entail any drawback since the resolution of threats, as potential conflicts they are, can be postponed in the problem-solving process. The solution to this threat is to insert the ordering constraint ($\gamma_2 \prec_\Pi \gamma_5$). The threat in Figure 2.46(a) is encompassed in the *action-argument* threats in DeLP-POP, and the threat in Figure 2.46(b) is included in the *argument-argument* threat definition of DeLP-POP.

Internal Support Threats (IST). The two situations that arise as an IST are depicted in Figures 2.46(c) and 2.46(d). In both cases, $k_1 \in AR(\Pi)$, $k_2 \in AA(\Pi)$ and the support (k_1, ℓ, k_2) is not of the form of a link but an internal literal n in the supporting argument $k_1 = \mathcal{B}$ that is necessary to derive the conclusion ℓ for $k_2 = \gamma_2$. In case 2.46(c), $k_3 = \gamma_3$ is the threatening step and the solution to this threat is to insert the ordering

constraint $(\gamma_2 \prec_\Pi \gamma_3)$. In case 2.46(d), the solution also involves the application of demotion through the ordering constraint $(\gamma_2 \prec_\Pi \gamma_5)$. The ISTs are also encompassed by the *action-argument* and *argument-argument* threats in DeLP-POP since this framework does not distinguish the support that is being threatened.

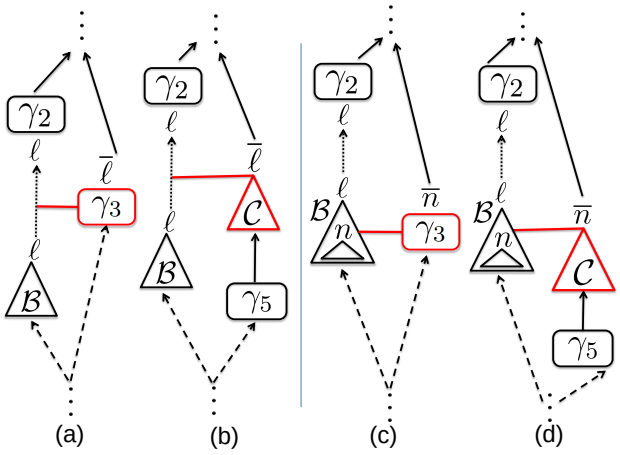

Figure 2.46: Examples of threats of type SLT and IST.

We must note that DeLP-POP introduces one more type of threat called *action-assumption*, in which an action threatens the base of an argument even though the base is not warranted yet (reason why the threaten is so-called *assumption*). However, under our threat definition, a threat does not exist until a support is explicitly introduced in the plan; that is, until the base of the argument is warranted via an action-argument, in which case it would become a CLT and would be solved by promotion or demotion. It has been proved that many potential threats can actually be delayed until the end (114) and in practice many planners adopt threat deferral as a flaw selection strategy (46, 114). The inclusion of supporting arguments in the threat machinery does not change this circumstance because the no-argument-supports-argument policy implies that any threatening or threatened supporting argument is involved in a threat as long as its base is supported by an action-argument.

Attacks. Attacks are conflicting situations that happen in a plan as a notice issued by an agent that some unexpected circumstances might affect the executability of an action

in the plan. Attacks are based on the contextual information or beliefs of the agents and they are detected so as to provide a stronger guarantee of success of the plan at execution time.

Formally, given a plan Π, we define an attack as a tuple $\langle \mathcal{A}_0, \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}, \Psi \rangle$, where:

- $\mathcal{A}_0$ is an argument of Π, either a supporting argument ($AR(\Pi)$) or a fictitious argument of an action-argument ($AA(\Pi)$).
- $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ is the set of attacking arguments such that $\mathcal{A}_1$ attacks $\mathcal{A}_0$, $\mathcal{A}_2$ attacks $\mathcal{A}_1$, and so on.
- Ψ is a set of literals that denote the activation context for the attacking arguments within the plan[1].

Notice that the first two elements of the attack tuple form an argumentation line $[\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_n]$ as defined in section 2.7.2.3.

Example 3. *Let's suppose an emergency case where the system demands a plan to move an ambulance from the hospital to the patient's home. An agent believes that the traffic congestion in the road that connects the hospital and the patient's home may prevent the ambulance from arriving in time. In this case, taking into account this contextual information in the design of the plan might be helpful to ensure the executability of the plan at runtime; that is, that the ambulance will get on time to the patient's home (27).*

Formally, given an agent whose knowledge base is $T = (\Psi, \Delta)$, it can build an argument $\mathcal{A}_1$ in T that attacks an argument $\mathcal{A}_0$ of a plan Π if the conclusion of $\mathcal{A}_1$ contradicts some literal derived in $\mathcal{A}_0$, that is, if $\overline{\mathsf{concl}(\mathcal{A}_1)} \in \mathsf{head}(\mathsf{Rul}(\mathcal{A}_0))$ (see definition of attack relation in section 2.7.2.1). Particularly, in Example 3, an agent builds an argument $\mathcal{A}_1$ which $\mathsf{concl}(\mathcal{A}_1)$ =*'ambulance not on time'* contradicts $\mathsf{concl}(\mathcal{A}_0)$ =*'ambulance on time'* given that both arguments use the same road between the hospital and the patient's home to make their inferences ($\mathsf{base}(\mathcal{A}_0) \cup \mathsf{base}(\mathcal{A}_1) \subseteq \Psi$); and the agent is aware of a congestion situation in such a road. This attack is represented as $\langle \mathcal{A}_0, \{\mathcal{A}_1\}, \Psi \rangle$ where: $\mathcal{A}_0$ is the argument of Π; $\mathcal{A}_1$ is the attacking argument; and, Ψ is the set of literals where $\mathsf{body}(\mathcal{A}_1)$

[1]The meaning of Ψ is the same as in the knowledge base $T = (\Psi, \Delta)$ of an agent but here it is referred to a particular context of the plan Π.

is warranted. Additionally, an argument $\mathcal{A}_2$ against $\mathcal{A}_1$ can be built in Example 3 if, for instance, an agent also holds an additional information of the existence of a fast track lane that avoids the traffic congestion. In this case, an argument $\mathcal{A}_2$ constructible in (Ψ, Δ) is a defeater of argument $\mathcal{A}_1$, such that the attack tuple now is formed by $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, \Psi \rangle$ and $[\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2]$ is the argumentation line. Note that attacking arguments $\mathcal{A}_1$ and $\mathcal{A}_2$ do not support any item in $\mathsf{goals}(\Pi)$ and they are agents' beliefs specifically used to attack or defend $\mathcal{A}_0$.

An initial attack of an argument $\mathcal{A}_1$ against an argument $\mathcal{A}_0$ of Π is represented in Figures 2.47(a), 2.47(b), 2.47(c) and 2.47(d). The four figures feature the attack $\langle \mathcal{A}_0, \{\mathcal{A}_1\}, \Psi \rangle$, where $\mathcal{A}_0$ is the fictitious argument of γ_1 in Figures 2.47(a) and Figure 2.47(d), and a supporting argument ($\mathcal{A}_0 \in AR(\Pi)$) in Figures 2.47(b) and Figure 2.47(c). Particularly, in Figure 2.47(b), $\mathcal{A}_1$ is attacking $\ell \in \mathsf{concl}(\mathcal{A}_0)$; and, in Figure 2.47(c), $\mathcal{A}_1$ is attacking n, an internal literal of $\mathcal{A}_0$. This different attacking point in the argument $\mathcal{A}_0$ does not make any difference in the semantics of the attack. Although it is not graphically represented in Figure 2.47, an argument $\mathcal{A}_2$ could attack $\mathcal{A}_1$, thus giving rise to the attack tuple $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, \Psi \rangle$.

It is important to note that all the attacking arguments $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ have to be activated in the same context Ψ of the plan Π; that is, $\mathsf{base}(\mathcal{A}_1)$, $\mathsf{base}(\mathcal{A}_2)$, ..., $\mathsf{base}(\mathcal{A}_n)$ have all to be warranted in Ψ, which in turn depends on the plan step of Π to which argument $\mathcal{A}_0$ is giving support (for instance, in Figure 2.47, argument $\mathcal{A}_0$ is giving support to the action-argument γ_2). The activation of the attacking arguments is explained in detail in the next section.

As shown in Figure 2.41(a) of section 2.7.2.1, given a root argument $\mathcal{A}_0$, the union of its argumentation lines gives rise to a tree-like structure, the dialectical tree for $\mathcal{A}_0$. In order to evaluate the root argument, the same procedure of Figure 2.41(a) is applied. At the end, the root argument is labeled as U or D. If $\mathcal{A}_0$ is labeled as U, it means there is no evidence against the conclusions of $\mathcal{A}_0$. Otherwise, agents will consider that there are reasons to believe that conclusions of $\mathcal{A}_0$ might not be achieved.

2.7.3.3 Activation of Attacking Arguments

The two attacking $\mathcal{A}_1$ and $\mathcal{A}_2$ of Example 3 shown in the previous section are constructible in (Ψ, Δ); that is, the base of the two arguments are to be warranted in Ψ. Therefore, we need to check that the base of the attacking arguments for or against an argument $\mathcal{A}_0$ of

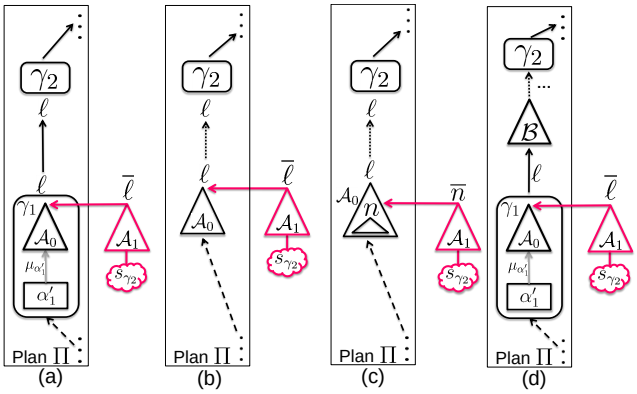

Figure 2.47: Examples of types of attacks.

Π are warranted in the state of the plan step to which $\mathcal{A}_0$ is giving support. That is, an attack or defense to $\mathcal{A}_0$ affects the plan step being supported by $\mathcal{A}_0$ and, consequently, the potential attacking arguments must be activated in the same context in which the corresponding plan step would be executed. In order to prevent an attacking argument from a false activation, we need to define a specific procedure to compute 'the state of a plan step'.

In state-based planning, a plan is a linear sequence of actions and the consistent state that holds before each action is known. However, a partial order plan Π is a set of actions whose execution ordering $\prec_\Pi$ is only partially specified, thus encoding multiple linear plans. Hence, POP does not explicitly represent state information associated to the actions in the plan.

Since states are not explicitly represented in POP, Q-DeLP-POP needs to calculate, for each $\gamma \in AA(\Pi)$, the possibly inconsistent set of literals potentially planned to occur before γ. Specifically, in (26), we presented and formalized the problem of identifying possible states in a partial plan with the notion of *proto-state*. Basically, the partial order of a plan Π determines, for each $\gamma \in AA(\Pi)$, a possibly inconsistent set of literals planned to occur before γ. The *proto-state* (also known as *Cutsets* in (79)) of an action-argument step γ can be seen as the set of literals that may hold before γ. The latest version of DeLP-POP (112) similarly introduced a concept named *propagated effects* for an action α, which is the set formed by adding together the effects of the actions that may be possibly ordered before α.

The set of *proto-states* of a plan must be continuously updated every time a step or an order constraint is inserted into the plan. As the plan search progresses, the *proto-states* will match better the sates that will result from the execution of the solution plan. Definition 7 presents formally the notion of *proto-state*.

Definition 7. **[Q-DeLP-POP Proto-State]**. *Let* Π *be a plan for* $\mathbb{M}$*, and* $\gamma, \gamma', \gamma''', \ldots \in AA(\Pi)$*. In* Q-DeLP-POP*, the proto-state of an action-argument step* γ *in* Π*, labeled as* s_γ*, is comprised by*[1]*:*

$$s_\gamma = \{\ell \in \mathsf{Lit} \mid \exists \gamma' \in AA(\Pi), \ell \in \mathsf{X}(\gamma') \text{ and } \prec_\Pi \cup \{\langle \gamma', \gamma \rangle\} \text{ is consistent, and } \forall \gamma'' \in AA(\Pi), \text{ if } \overline{\ell} \in \mathsf{X}(\gamma'') \text{ then } \{\langle \gamma', \gamma'' \rangle, \langle \gamma'', \gamma \rangle\} \nsubseteq \mathsf{tc}(\prec_\Pi \cup \{\langle \gamma', \gamma \rangle\})\}$$

Figure 2.48 shows an example of a partial plan with five action-argument steps. The *proto-state* of the action-argument γ_5, i.e., the set of literals that can possibly occur before γ_5 is $s_{\gamma_5} = \{\overline{p}, q, \overline{q}\}$, where: $\overline{p}$ is part of s_{γ_5} due to the ordering constraint $(\gamma_2 \prec_\Pi \gamma_5)$; p is not included in s_{γ_5} because of the ordering constraint $(\gamma_1 \prec_\Pi \gamma_2)$ and $\mathsf{X}(\gamma_2)$ denies $\mathsf{X}(\gamma_1)$; and, q and $\overline{q}$ are part of s_{γ_5} because $\langle \gamma_3, \gamma_5 \rangle$ and $\langle \gamma_4, \gamma_5 \rangle$ are possible relations consistent with $\mathcal{OC}(\Pi)$. Following Definition 7, we observe that $\langle \gamma_3, \gamma_4 \rangle$ belongs to the transitive closure of the relations in the plan but $\langle \gamma_4, \gamma_5 \rangle$ does not, reason why $q \in s_{\gamma_5}$. Note that Figure 2.48 shows an example of a *proto-state* containing an inconsistent set of literals.

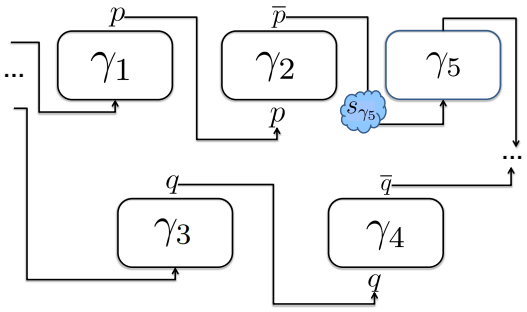

Figure 2.48: Example of the *proto-state* $s_{\gamma_5} = \{\overline{p}, q, \overline{q}\}$.

Back to Figure 2.47, the base of the attacking argument $\mathcal{A}_1$ needs to be warranted in the proto-state of the action-argument γ_2. This is graphically shown in Figure 2.47 by a

[1] We use tc to refer to the transitive closure

cloud drawn under $\mathcal{A}_1$. Therefore, the attack tuple $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, \Psi \rangle$ ($\mathcal{A}_2$ is not graphically represented) is now replaced by $\langle \mathcal{A}_0, \{\mathcal{A}_1, \mathcal{A}_2\}, s_{\gamma_2} \rangle$ in order to ensure that the attacking arguments against $\mathcal{A}_0$ are warranted in the proto-state of γ_2, the action-argument supported by $\mathcal{A}_0$. Additionally, in Figure 2.47(d), we have that $(\gamma_1, \ell, \mathcal{B}) \in \mathcal{CL}(\Pi)$ and $(\mathcal{B}, z, \gamma_2) \in \mathcal{SL}(\Pi)$. In this case, the proto-state or activation context of the attacking argument $\mathcal{A}_1$ would also be s_{γ_2}. The reason is that an attacking argument represents a belief of an agent against a successful execution of an action so, ultimately, the target of an attacking argument is always an action-argument step. This is also confirmed by Definition 7, where proto-states are only formed by the effects of the action-argument steps, and hence γ_1 would be indirectly supporting γ_2 in this example.

2.7.4 Cooperative Planning Protocol

Unlike DeLP-POP, where a single agent is responsible of the argumentation-based planning process, Q-DeLP-POP has been extended to a multi-agent context. In this section, we present the application of Q-DeLP-POP to a Multi-Agent Planning (MAP) scenario, hereinafter labeled as Q-DeLP-MAP.

2.7.4.1 Multi-Agent Planning Task

Agents in Q-DeLP-MAP are entities of the problem equipped with planning and argumentation capabilities that possess their own planning and contextual information. Therefore, given a planning task $\mathbb{M}$, each agent will have a different and local view of the task, $\mathbb{M}_{Ag_i} = \langle \Psi_{\mathsf{Ag}_i}, \Delta_{\mathsf{Ag}_i}, A_{\mathsf{Ag}_i}, G \rangle$.

We denote a planning team as $\mathsf{AG} = \{\mathsf{Ag}_1 \dots \mathsf{Ag}_n\}$ such that $n = |AG|$ is a finite non-empty set of agents with planning and argumentation capabilities. It is assumed that agents are fully cooperative. Each agent Ag_i is endowed with a tuple $\mathbb{M}_{Ag_i}$, where:

- $\Psi_{\mathsf{Ag}_i} \subseteq \Psi$ represents the partial view of the initial state of agent Ag_i such that $\Psi = \bigcup_{\forall \mathsf{Ag}_i \in \mathsf{AG}} \Psi_{\mathsf{Ag}_i}$ is a consistent set.
- $\Delta_{\mathsf{Ag}_i} \subseteq \Delta$ is the set of defeasible rules known by agent Ag_i such that $\Delta = \bigcup_{\forall \mathsf{Ag}_i \in \mathsf{AG}} \Delta_{\mathsf{Ag}_i}$ is a set of possibly contradictory rules.
- $A_{\mathsf{Ag}_i} \subseteq A$ is the set of planning actions known by agent Ag_i such that $A = \bigcup_{\forall \mathsf{Ag}_i \in A} A_{\mathsf{Ag}_i}$.

- G is a set of global goals that represent the needs of a user in an environment. Unlike the rest of elements, the goal state G is known to all the agents.

Q-DeLP-MAP is aimed at cooperative planning tasks where agents contribute to creating the plan with their planning actions and to ensuring the executability of the plan through their defeasible rules. Thus, agents in a MAP task jointly solve G and help discover the non-anticipating conditions that might prevent the plan from being executable.

The team formation is not covered in this article and we assume that AG is a static team that does not vary during the planning process. The formation of dynamic teams, where agents autonomously enter and exit the team, falls within the investigation in open Multi-Agent Systems (MAS) and it will addressed in future works.

A relevant aspect in MAS is the notion of privacy. Let $\mathbb{M}_{Ag_i} = \langle \Psi_{\mathsf{Ag}_i}, \Delta_{\mathsf{Ag}_i}, A_{\mathsf{Ag}_i}, G \rangle$ and $\mathbb{M}_{Ag_j} = \langle \Psi_{\mathsf{Ag}_j}, \Delta_{\mathsf{Ag}_j}, A_{\mathsf{Ag}_j}, G \rangle$ be the planning task as regarded by two agents Ag_i and Ag_j:

- Ag_i and Ag_j can share some of the planning capabilities, in which case $A_{\mathsf{Ag}_i} \cap A_{\mathsf{Ag}_j} \neq 0$.
- Ag_i and Ag_j can share all or some of the facts in the initial world state, in which case $\Psi_{\mathsf{Ag}_i} \cap \Psi_{\mathsf{Ag}_j} \neq 0$.
- Ag_i and Ag_j can share some of their beliefs, in which case $\Delta_{\mathsf{Ag}_i} \cap \Delta_{\mathsf{Ag}_j} \neq 0$.
- Ag_i and Ag_j share the set G as these are the common goals to be cooperatively achieved by both agents.

If a literal l is private to agent Ag_i then l is not shared with any other agent and l can only be used by Ag_i during the plan construction and argumentative evaluation. The more private information of the agents, the less the interaction between them during the planning and argumentation. For this reason, in Q-DeLP-MAP scenarios, it is important that as much information as possible is labeled as public in order to promote argumentation as a coordination mechanism that enable agents to contrast their beliefs about the world towards a successful achievement of an executable plan.

Specifically, the existence of public information is more concerned with Ψ, the initial world state, and the successive planning states generated along the plan construction. However, it is commonly accepted that two agents may have different defeasible rules; for instance, Ag_i may possess information that allows the agent to infer the state of the traffic whereas Ag_j may have defeasible information about the weather. Similarly, it is

also commonly accepted that in a planning task an agent represents an entity with different planning capabilities than other agents; this is the case, for instance, when Ag_i represents a *plane* and Ag_j represents a *truck* in a logistic domain. Note also that in Q-DeLP-MAP planning data as well as beliefs of an agent denote defeasible information since, unless they involve some private data to the agent, it can be rebutted by the rest of the agents.

Multi-Agent Planning is required when the domain of application is composed of multiple entities that are distributed *functionally*, in which case they need to work together to solve the planning task, or *spatially*, in which case they accomplish the planning task better by cooperating (115). On the other hand, distributed execution promotes the efficiency of parallel processing of actions, the robustness of the system to cope with complex planning problems and the scalability of a construction across a network of interconnected agents (each agent is installed in a different machine), thus avoiding the critical failures and resource limitations of centralized systems.

2.7.4.2 Multi-Agent Search Protocol

Figure 2.49 outlines the three main stages of the Q-DeLP-MAP protocol: *plan and goal selection*, *plan generation* and *plan evaluation*. Given a planning task $\mathbb{M}$, and a set of agents AG, the Q-DeLP-MAP protocol starts with an initial empty plan, $\Pi_0 = \{\alpha_\Psi \prec \alpha_G\}$, and agents progressively search through a space of possible plans (POP tree). At each iteration of the protocol, agents collaboratively select a node of the POP tree and expand it. The process finishes when a plan in which all step preconditions are necessarily true is found (solution plan). The final goal of Q-DeLP-MAP is to return a solution plan with guarantees of robust execution in the real-world.

Unlike other approaches, Q-DeLP-MAP successively interleaves planning and argumentation so as to build a plan incrementally. Once a plan Π is selected by the agents, and unless Π is labeled as a solution plan, agents select an open goal Φ of Π. Then, Π is expanded in the plan generation stage, where agents put forward and exchange refinement plans of Π that would potentially solve Φ. Following, agents evaluate each plan proposal by argumenting the unexpected circumstances that might occur in the context of the proposal. Plan evaluation consists of two steps; (1) the application of the *Case-based Reasoning* module in order to find similar argumentative cases and (2) the *Plan Argumentation* phase, where agents get involved in an argumentative dialogue in which they expose their arguments for or against the plan proposals.

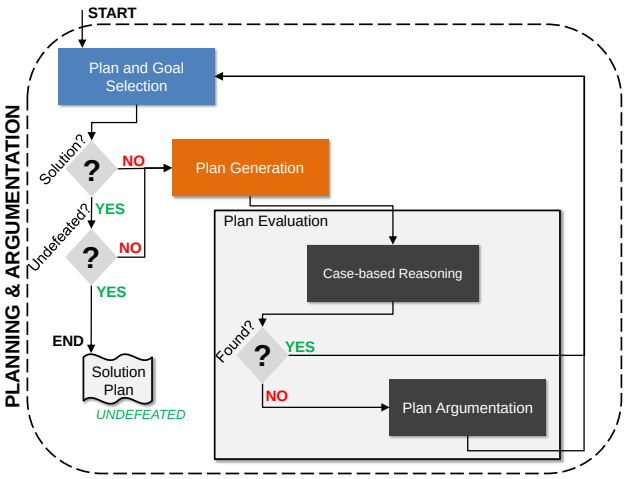

Figure 2.49: Main algorithm of Incremental Plan Construction in Q-DeLP-MAP.

Plan generation Plan refinements are the plan proposals put forward by the agents as a result of expanding a selected plan Π when solving an open goal Φ. This stage follows a process similarly to a plan-space planning process that builds a POP tree, except that each refinement or successor of Π can be now generated by a different agent and can contain arguments to support the action preconditions or arguments base. Agents exchange their refinements to each other and learn the new steps, literals and links of the plan, which they keep in their local POP tree.

In a previous multi-agent version of DeLP-POP (26), agents engage in a turn-based dialogue permitting agents to collaborate to discover threats to any argument step of a refinement. However, in Q-DeLP-MAP, as explained in section 2.7.3.2), threats caused by an argument step $\mathcal{A}$ are not solved until $\mathsf{base}(\mathcal{A})$ is supported, thus permitting any other agent of AG to insert action-arguments in the plan to support $\mathsf{base}(\mathcal{A})$ and solve the threat by promotion or demotion. This is consistent with the partial-order structure of the plan and, consequently, with the local information held by the agents. Subsequently, agents will learn the new ordering constraints and will update their local POP tree accordingly. Additionally, unlike DeLP-POP (26), in Q-DeLP-MAP the refinements of Π generated by the agents are free of threats. It means that if an agent finds a threat in one of its new refinements, then it solves it before exchanging the new proposal with the rest of the

agents.

Agents need to coordinate the exchange of their plan proposals. In Q-DeLP-MAP, the coordination protocol is based on a *democratic leadership* where a leadership baton is scheduled among the agents following a round-robin strategy. Once the baton agent has sent his refinements to the rest of the agents, and acknowledgment is received, the coordination stage is completed, and the baton is handed over to the following agent. Thus an agent can only speak when it holds the baton, and it is listening the rest of the time. The process is repeated until all the agents have once taken the baton role.

Plan evaluation Evaluating a plan and analyzing its executability in the real world, according to the agents' beliefs, is done through two alternative protocols: *Plan Argumentation* and *Case-Based Reasoning (CBR)*. In the *Plan Argumentation* stage, agents become engaged in a series of argumentative dialogues aimed at evaluating the guarantee of a successful execution of a plan proposal. Specifically, argumentation occurs in the context of the agents' beliefs and they build arguments to defend or attack a plan proposal. Subsequently, the CBR will register the argumentative case in order to be able to re-use it in further evaluations.

Given a refinement or plan proposal Π_r, agents generate as many argumentative dialogues as supporting arguments and action-argument steps are present in Π. Let $\mathcal{A}^{\mathsf{Ag}_i}$ be one argument of Π_r, where Ag_i is the agent that inserted $\mathcal{A}$ in Π_r, and let γ be the action-argument to which $\mathcal{A}$ is giving support. The argumentative dialogue to evaluate $\mathcal{A}^{\mathsf{Ag}_i}$ is encoded as a dialectical tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$ (see Section 2.7.2.1). Nodes of $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$ are labeled with an argument that attacks the argument in its parent node and which base is supported in the proto-state s_γ. More specifically:

1. The root node of the tree is labeled with $\mathcal{A}^{\mathsf{Ag}_i}$ such that $\mathcal{A}^{\mathsf{Ag}_i} \in \mathcal{AR}(\Pi_r)$ or $\langle \alpha', \mathcal{A}^{\mathsf{Ag}_i} \rangle \in AA(\Pi_r)$. The argumentative process identifies the action-argument γ of Π supported by $\mathcal{A}^{\mathsf{Ag}_i}$ and creates the corresponding proto-state, s_γ.

2. A child node $\mathcal{B}^{\mathsf{Ag}_j}$ of $\mathcal{A}^{\mathsf{Ag}_i}$ represents an attacking argument against $\mathcal{A}^{\mathsf{Ag}_i}$; i.e., $\mathcal{B}^{\mathsf{Ag}_j}$ is a defeater of $\mathcal{A}^{\mathsf{Ag}_i}$. Consequently, children of $\mathcal{A}^{\mathsf{Ag}_i}$ stand for defeaters of the root argument $\mathcal{A}^{\mathsf{Ag}_i}$.

3. A child node $\mathcal{C}^{\mathsf{Ag}_z}$ of $\mathcal{B}^{\mathsf{Ag}_j}$ indicates an attack against $\mathcal{B}^{\mathsf{Ag}_j}$, so this new node is actually a supporter of the root argument $\mathcal{A}^{\mathsf{Ag}_i}$.

4. And so on.

The above process creates an argumentation line $\langle \mathcal{A}^{\mathsf{Ag}_i}, \{\mathcal{B}^{\mathsf{Ag}_j}, \mathcal{C}^{\mathsf{Ag}_z}, \ldots\}, s_\gamma \rangle$ of $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_i}}$. A dialectical tree is generated for each argument in Π_r and the leaves of the tree are undefeated arguments. Unlike the dialectical trees of a general argumentative process (24), the nodes in our dialectical tree are supported in the proto-state of the action-argument supported by the root argument. Every linear path from the root to a leaf corresponds to one different acceptable *argumentation line*. Circular argumentation (also known as fallacious argumentation) is avoided by applying both conditions from (24): no argument can be reintroduced in the same argumentation line and argument concordance must be guaranteed.

Similarly to the *Plan Generation* phase, agents can only adopt one role, namely, the baton agent, or the participant agent, during the argumentative process, and the roles are iteratively exchanged. The baton agent is the agent that inserted the argument in Π_r and it is the responsible for the construction of the dialectical tree. The baton agent is also allowed to put forward a self-attacking argument as well as processing the attacking/supporting arguments of the rest of agents in the dialectical tree. The *Plan Argumentation* phase of a particular argument ends when an argumentative round with no new attacking arguments is detected.

The *Plan Argumentation* phase is applied to every argument in all nodes of a POP search tree. Let Π_r be a node which is expanded into a child node Π'_r. Many of the arguments in Π'_r will also be comprised in Π_r and will have been evaluated at the plan argumentation stage of Π_r. Since the structure of a partial plan changes along its construction, the proto-state of some action-arguments in Π'_r may have changed and others may have not with respect to Π_r. Thus, it seems convenient to store the results of the *Plan Argumentation* phase so as to re-use them in the evaluation of refinement successors.

Case-Based Reasoning (CBR) systems allow agents to learn from their experiences (62). Q-DeLP-MAP uses a *CBR* module with the only objective of improving the performance of the plan evaluation stage. A case in the argumentation database is a 4-tuple structure $\langle \mathcal{A}, \mathcal{T}_\mathcal{A}, s_\gamma, r \rangle$, where $\mathcal{A}$ is the root argument, $\mathcal{T}_\mathcal{A}$ is the dialectical of $\mathcal{A}$; s_γ is the proto-state of γ, the action-argument directly or indirectly supported by $\mathcal{A}$; and, r is the result of the evaluation of $\mathcal{A}$, that is, D (*defeated*) or U (*undefeated*). This way, whenever an argument is evaluated, a new 4-tuple case is inserted in the argumentation database. Likewise, before applying the *Plan Argumentation* to an argument, we perform

a query to the database to search for a similar past experience. If a similar case is found, the *Plan Argumentation* protocol is not executed and the step is directly labeled with r. Otherwise, a new argumentation phase is launched. In this way, Q-DeLP-MAP avoids to repeat similar *Plan Argumentation* protocol. As we will see in section 2.7.5, using the *CBR* module allows for a more efficient model and solve more complex problems than the ones in (27, 28).

Plan and Goal Selection phase In order to select the next open goal from $\mathsf{goals}(\Pi)$, Q-DeLP-MAP applies a heuristic function that selects the most costly goal according to a reachability analysis method based on the relaxed planning graph (RPG) of the planning task (116). Specifically, the cost of a goal g, $cost(g)$, is estimated as the first literal level of the RPG where g appears. This is clearly an underestimation of the real cost of g since $cost(g)$ only accounts for the number of actions necessary to reach g from the initial situation with no regard of the cost of the actions preconditions or the possibly negative interactions among actions in the plan.

As for the selection plan heuristic, Q-DeLP-MAP applies first a warranty procedure to discard the plans evaluated as *defeated* in the plan argumentation phase. Subsequently, the *undefeated* plans are estimated according to $h(\Pi) = cost(\mathsf{goals}(\Pi))$ (79); specifically, $h(\Pi)$ is estimated as $h(\Pi) = \sum_{g \in \mathsf{goals}(\Pi)}(cost(g))$. The possibly overestimation of this additive heuristic lies in that one same action that achieves the precondition of two or more actions is counted as many times as needer actions. In contrast, this overestimation balances out the underestimation of $cost(g)$.

2.7.5 Experimental evaluation

Q-DeLP-MAP draws upon CAMAP (27, 28), a preliminary implementation of the argumentation-based MAP model specifically adapted to applications of ambient intelligence in the field of health-care. CAMAP was only able to solve simple planning problems. Q-DeLP-MAP, however, is a domain-independent framework which has been optimized to address more complex planning problems, incorporating features such as the CBR module. In order to analyze the benefits and limitations of Q-DeLP-MAP, we compare it with two other MAP approaches:

- MAP-POP: a general-purpose MAP framework suitable to cope with a wide variety of multi-agent planning domains (33). MAP-POP is a POP-based refine-

ment planning approach that iteratively combines planning and coordination where agents have only the ability of planning; i.e. no argumentation is used in MAP-POP.

- PS-Q-DeLP-MAP: we also tested Q-DeLP-MAP for plan selection (PS) instead of applying defeasible reasoning during the search process of building a plan. In this case, the plan generation phase is executed until completion (a solution plan that solves the problem goals is returned) and then the plan is evaluated in an argumentative dialogue among the agents. PS-Q-DeLP-MAP emulates the behavior of other approaches (29, 30) that draw upon a one-shot planning-argumentation approach instead of continuously interleaving planning and argumentation like in Q-DeLP-MAP. This iterative one-shot procedure is repeated until the evaluation phase returns an undefeated solution plan.

The experiments were carried out in two well-known domains used in the IPC (International Planning Competitions):

- rovers: it is a simplification of the NASA Mars Exploration Rover problem. Multiple planetary rovers explore the environment by taking pictures, gathering samples and communicating them back to a lander. The rovers benchmark includes problems where all rovers have typically capabilities for gathering samples and only one rover is equipped with a camera to take pictures.
- logistics: it involves driving trucks and airplanes around delivering packages between locations. Locations are either airports, only reachable by planes, or places within a city, only reachable by trucks. The objective is to deliver packages to their destinations, which can be any location in the problem.

Since the IPC problem suites only contain single-agent versions of the planning problems, we used the multi-agent version of the rovers and logistics domains presented in (115)[1]. The agentization of the rovers domain consists in creating a planning task per rover. In this case, agents are all of the same type (rovers) and, therefore, the same set of planning actions is defined in all of the tasks. However, two different types of agents are identified in the logistics domain, trucks and planes, thus generating a *truck* or a *plane* planning task per agent. On the other hand, the rovers domain generates loosely-coupled tasks because rovers are generally able to achieve a problem goal by themselves although

[1]Domains are also available at http://users.dsic.upv.es/grupos/grps/tools/map/fmap.html

they need to access certain shared resources in their environment, namely the rock and the soil samples they collect and analyze. In contrast, the logistics problems fall into the tightly-coupled category since agents need to cooperate in order to transport the packages to their target locations and tasks present several coordination points (airports) at which trucks and planes can interact (115).

Specifically, the planning problems in (115) are encoded in PDD3.1 language[1], a version of PDDL language that introduces state variables. We then incorporated the defeasible rules into the planning tasks of the agents[2]. In the rovers domain, defeasible rules are related to:

1. due to the existence of solar storms, one or more agents may believe this is a probably cause of communication failures between a rover and the lander whereas other agents may believe the opposite.

2. agents that have a more accurate picture of the mars surface may hold reasons to believe that a rover will not be able to move across some particular area at night. However, some rovers may be equipped with a spotlight, which would allow them to move between waypoints at night, being this information is unknown to the attacking agent.

3. an agent may hold reasons to believe that the usual low temperatures in the space will prevent a rover from moving to a specific waypoint; but, if the rover is equipped with a electrical heater, he would be able to refute the attack of the agent.

We must note that news like the solar storms or atmospheric conditions in the space may be likewise considered as defeasible knowledge if we assume that the agent might have an outdated information. This frequently happens in a Mars domain since current Mars rovers communicate to Earth via orbital relays and the communication from Mars to Earth has a long delay of at least 2.5 minutes, and at most 22 minutes. Consequently, depending on the time elapsed since the news was received from Earth, it can be managed as a fact or a belief.

[1] http://ipc.informatik.uni-freiburg.de/PddlExtension

[2] Defeasible rules are encoded in the form of operators, like planning operators, through the special constructor :def-rule (see (28)) of our language. Internally, our planner works with ground instances of operators like most planners do. Ground instances of :def-rule are generated using the set of literals derived from the grounding of the planning operators plus the literals of the initial situation. In the following, we will use here the term defeasible rule to refer to any instance of a :def-rule operator.

	Problem 1 (R-Pfile1 and L-Pfile1)				Problem 2 (R-Pfile3 and L-Pfile3)				Problem 3 (R-Pfile4 and L-Pfile4)				Problem 4 (R-Pfile5 and L-Pfile5)				Problem 5 (R-Pfile7 and L-Pfile6)			
Problems	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**	**Ag**	**Act**	**Def**	**Go**
Rovers Simple	1R	41	25	3	2R	52	25	3	2R	86	44	3	2R	144	44	7	3R	151	71	6
Rovers Hard	1R	41	64	3	2R	52	64	3	2R	86	123	3	2R	144	123	7	3R	151	161	6
Logistics Simple	1P 2T	56	31	4	1P 2T	80	31	6	1P 3T	133	39	7	1P 3T	156	39	8	1P 3T	174	81	9
Logistics Hard	1P 2T	56	62	4	1P 2T	80	62	6	1P 3T	133	87	7	1P 3T	156	87	8	1P 3T	174	164	9

Figure 2.50: Configuration of the Experiments.

In the logistics domain, defeasible rules are concerned with the following knowledge held by agents:

1. weather conditions: some agents may have a more precise weather forecast or simply know that bad weather conditions may provoke delays in takeoffs or even flight cancellations.

2. environment conditions: a relatively common situation that may alter the elaboration of any plan is a call for a transport strike. Initially, this is handled as a prediction, later either it is confirmed or suspended. Agents can then argue about the possibility that the strike takes place and the consequences in the transport plan.

We created two defeasible scenarios, simple and hard, for all the problems in each domain (see Figure 2.50). Basically, the difference between these two scenarios relies in the number of defeasible rules that contradict to each other. The overall number of defeasible rules used in each problem can be seen in Figure 2.50 under the legend 'Def'. Particularly, the additional defeasible rules of the hard problems are designed in such a way that their heads contradict the head of some other rule of the respective simple problem. Hard scenarios introduce a higher level of disagreement among agents but this does not necessarily entail a higher level of defeated arguments since more attacks to arguments also imply more rebuttals and, consequently, this may lead to more supports. Defeasible rules were distributed across agents independently of the agent type, thus making agents be able to extract context inferences about any issue of the context. A detailed description of the specification of the defeasible rules can be found in the paper that describes CAMAP (28).

We tested five problems for each domain, which all correspond to a particular problem of the IPC suites. In the IPC, problems are identified with names pfile1, pfile2, etc., where

it is usually the case that the higher the file index, the more complex the problem is[1]. The correspondence between our problems and the IPC pfiles is shown in Figure 2.50 under the caption of each problem. In both domains, Problem1, Problem2, Problem3 and Problem4 correspond to pfile1, pfile3, pfile4 and pfile5, respectively. Problem5 is the pfile7 of the rovers domain and pfile6 in the logistics domain.

The remainder values of Figure 2.50 indicate the size and complexity of the problems. The column 'Ag' shows the number of agents (R stands for rovers, T for trucks and P for planes). 'Act' is the number of planning actions and 'Go' is the number of goals to solve in the problem. As we can see, problems increase in complexity with respect to almost all the parameters: agents, actions, defeasible rules and goals.

All the agents are executed on a single machine with a 2.83 GHz Intel Core 2 Quad CPU and 8 GB RAM (only 1 GB RAM available for the Java VM); and a broker of Java Message Service[2] to allow the communication between agent is executed in other machine also with 2.83GHz and 1GB of RAM.

2.7.5.1 Performance analysis

The results obtained for the two scenarios (simple and hard) of the rovers domain are presented in Figures 2.51(a) and 2.51(b), respectively; and Figures 2.52(a) and 2.52(b) show the results for the logistics domain. Additionally, we present some figures of the complexity of the search process for the problems of the hard scenarios in Figure 2.53. We will use all these figures to analyze the performance of the three MAP approaches (MAP-POP, Q-DeLP-MAP and PS-Q-DeLP-MAP).

There are several performance parameters that determine the obtained results for the three approaches:

1. The size of the search space. This factor, which is directly related to the size of the problem (see Figure 2.50), affects the three approaches.

2. Number of planning messages exchanged between agents. This parameter also determines the performance of the three approaches and it is related to the size of

[1] We chose the pfiles so as to solve a fairly broad range of problems with different complexity and different number of agents

[2] MAP-POP, Q-DeLP-MAP and PS-Q-DeLP-MAP are based on Magentix2: a platform of multi-agent systems (117)

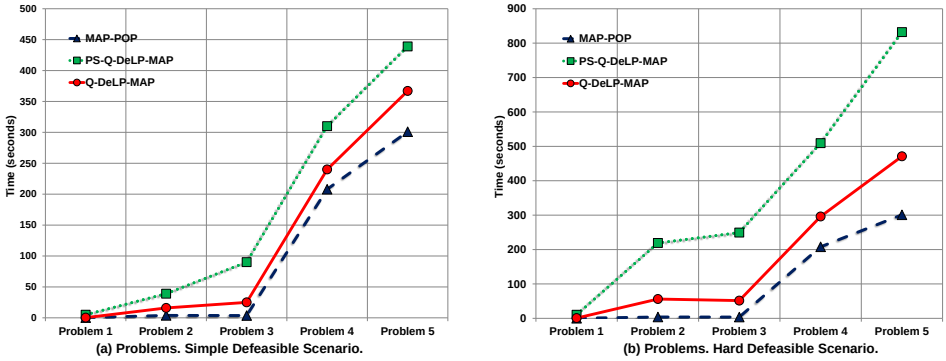

Figure 2.51: Evaluating the average time spent on each model by executing the rovers problem.

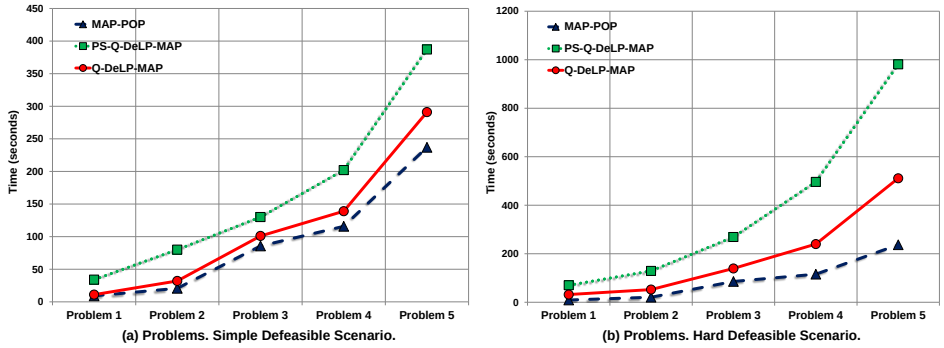

Figure 2.52: Evaluating the average time spent on each model by executing the logistics problem.

	MAPOP					Q-DeLP-MAP					PS-Q-DeLP-MAP				
Hard Defeasible Scenario	**P1**	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P5**
Rovers: Total number of proposed plans	1	30	20	450	504	1	386	22	664	606	3	9	9	18	26
Rovers: Total Number of Dialogues	0	0	0	0	0	10	341	60	2742	2632	24	90	63	360	442
Rovers: Arguments of CBR with Reuse = 0	0	0	0	0	0	10	144	35	1781	1583	20	71	55	325	412
Rovers: Arguments of CBR with Reuse ≥ 1	0	0	0	0	0	0	197	25	961	1049	4	19	8	35	32
Logistics: Total number of proposed plans	65	128	101	354	253	132	183	1366	557	1341	8	10	12	24	47
Logistics: Total Number of Dialogues	0	0	0	0	0	252	306	2562	1232	1363	144	230	396	648	1598
Logistics: Arguments of CBR with Reuse = 0	0	0	0	0	0	147	176	1355	669	819	102	192	321	387	1147
Logistics: Arguments of CBR with Reuse ≥ 1	0	0	0	0	0	105	130	1207	563	544	41	38	75	261	451

Figure 2.53: Results of the search process for the problems in the hard defeasible scenarios.

the problem as well as to the complexity of the domain. In the case of the tightly-coupled problems of the logistics domain, the high interaction level among agents

also entail a greater number of planning messages (plan proposals).

3. Number of dialogues or dialectical trees. This parameter only affects Q-DeLP-MAP and PS-Q-DeLP-MAP since no argumentation is employed in MAP-POP.

4. Number of argumentation messages exchanged between agents. This parameter only impacts in Q-DeLP-MAP because the argumentative dialogues (plan evaluation) take places along with the plan generation during the construction of the plan. However, the argumentation phase in PS-Q-DeLP-MAP is performed at once at the end of the plan generation, that is, over the solution plan.

5. Number of times the planning process (plan generation) is executed in PS-Q-DeLP-MAP. This parameter, which only affects PS-Q-DeLP-MAP, is related to the implementation of this approach. We used Q-DeLP-MAP as an external planner (with the planning evaluation stage deactivated) in order to obtain the solution plans for PS-Q-DeLP-MAP. Thus, we obtain a solution plan composed of action-argument steps that is sent to the argumentative process. If the plan turns out to be defeated, Q-DeLP-MAP is invoked again from scratch and the CBR module filters out the already discarded solution plans. This behaviour emulates exactly the usage of an external planner as in (29, 30).

Regarding these parameters, the values in Figure 2.53 give an idea of the complexity of each approach in the hard scenarios of both domains (P1 stands for Problem1, P2 stands for Problem2 and so on):

1. The number of proposed plans in MAP-POP and Q-DeLP-MAP is the total number of nodes (partial-order plans) generated in the search tree. Note that Q-DeLP-MAP sends all the generated nodes to the argumentative process so as to discard the defeated plans whereas MAP-POP only expands the nodes that result from the application of the heuristic function. Typically, Q-DeLP-MAP will generate more partial plans than MAP-POP since a node that is expanded by MAP-POP may be labeled as 'defeated' in Q-DeLP-MAP and so the node will be discarded. On the other hand, in PS-Q-DeLP-MAP the number of proposed plans indicates the total number of solution plans, that is, how many times Q-DeLP-MAP is invoked as an external planner (with no argumentative process activated).

2. The number of dialogues of MAP-POP is obviously 0. We remark that the difference in the number of dialectical trees between Q-DeLP-MAP and PS-Q-DeLP-MAP is due to their different behaviour as explained above: agents do not exchange argumentative messages in PS-Q-DeLP-MAP because the evaluation stage is applied only at the end of the plan generation.

3. The next two rows in each domain show the number of arguments that have never been reused or have been reused at least once out of the total number of evaluated arguments (dialogues).

We examine now the results of Figures 2.51, 2.52 and 2.53. The first observation on the results of Figures 2.51 and 2.52 is that MAP-POP is the most efficient approach thanks to the absence of an argumentative process. We can also see in Figures 2.51 and 2.52 that PS-Q-DeLP-MAP is always more costly than Q-DeLP-MAP although the number of proposed plans, dialectical trees and, consequently, the number of exchanged messages between agents is significantly higher in Q-DeLP-MAP than in PS-Q-DeLP-MAP (actually, there are no argumentation messages exchanged between agents in PS-Q-DeLP-MAP). This is explained as follows. In Q-DeLP-MAP, the argumentative process occurs at each node of the search tree where a successor node n' of a node n only introduces a new action-argument with respect to n. If the proto-states of the plan in n do not change in n' then the plan evaluation stage of n' will reuse all the dialectical trees of the parent node and only the new action-argument is sent for evaluation. However, the CBR module in PS-Q-DeLP-MAP can only reuse argumentative cases of solution plans, thus being necessary to generate a complete plan before discovering whether the solution plan is defeated or undefeated. This evidences the high computational cost of the planning machinery compared to the cost of the argumentation. Thus, as shown in Figures 2.51 and 2.52, interleaving planning and argumentation approach is more beneficial because argumentation helps conduct the planning process and reduce the planning workload even at the cost of a higher argumentation activity in the system.

With respect to the results of the rovers domain (Figures 2.51(a) and 2.51(b)), we can observe that the computation time of Problem2 and Problem3 is almost identical for the three approaches, particularly in the hard defeasible scenario. This is explained because the size of Problem2 and Problem3 is very similar, same number of agents and goals (see Figure 2.50), as it also reveals the similar number of proposed plans of MAP-POP and PS-Q-DeLP-MAP for these two problems. However, the number of proposed

plans and dialogues of Problem2 is significantly higher than Problem3 in Q-DeLP-MAP and even so the computation time is identical (Figure 2.51(b)). This reinforces the idea that the planning workload is still more determinant than the argumentation load in the performance of the model. We can also observe the increase in the computation time of Problem4 and Problem5. This is due to the increase in the number of goal of Problem4 (7 goals) with respect to Problem3 (3 goals), which is also translated into a significantly higher number of proposed plans and dialogues in all the approaches.

Regarding the logistics domain, Figures 2.52(a) and 2.52(b) show an almost linear increase in the computation time of the five problems. This is partly explained because, unlike the rovers domain, the size of the logistics problems increase gradually (see Figure 2.50) and so it does the size of the search space. However, the differences between the simple and hard defeasible scenarios are more pronounced in this domain and this affects particularly negatively to PS-Q-DeLP-MAP. In logistics, agents (trucks and planes) need each other to accomplish the goal so typically a plan will contain at least a truck and a plane. The more agents involved in the plan, the more attacks the plan will receive since defeasible rules are uniformly distributed across agents.

In summary, we can conclude that the performance of Q-DeLP-MAP pays off the utilization of the argumentation in solving multi-agent cooperative planning problems with respect to the other two MAP approaches.

2.7.5.2 Quality of the solution plans

In this section, we analyze the quality of the solution plans returned by the three MAP approaches. Figure 2.54 shows the number of actions and the duration (number of time or execution steps) of the solution plans for the five problems of the hard defeasible scenarios in both domains. By actions we mean the actions of the plan that are executable in the real-world where each of them corresponds with an action-argument step in Q-DeLP-MAP and PS-Q-DeLP-MAP.

The first observation is that the plans returned by Q-DeLP-MAP and PS-Q-DeLP-MAP are the same. The reason is that the planning model of PS-Q-DeLP-MAP is the same that Q-DeLP-MAP (without argumentative process activated); and, the way to build dialogues in PS-Q-DeLP-MAP is similar than Q-DeLP-MAP. The only difference is the way in which both models reach the solution plan.

We can also observe that the number of actions in Q-DeLP-MAP and PS-Q-DeLP-

	MAPOP					Q-DeLP-MAP / PS-Q-DeLP-MAP				
Quality Metrics of the Solution Plan	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
Rovers: Action Steps	10	12	8	24	18	8	10	7	20	17
Rovers: Duration	7	8	5	10	7	6	7	4	9	7
Logistics: Action Steps	20	25	37	31	36	18	23	33	27	34
Logistics: Duration	9	9	13	11	13	8	9	13	11	13

Figure 2.54: Quality of the solution plans for the hard scenarios.

MAP is always lower than in MAP-POP. The reason is that, unlike MAP-POP, in PS-Q-DeLP-MAP and Q-DeLP-MAP, an open goal of an action-argument can be supported through a supporting argument rather than with another action-argument; that is, the agents beliefs that succeed the argumentative process are used as a support of the conditions of the planning actions. For instance, let's assume a rover situated in waypoint w1 that plans to move to a waypoint w2 in order to collect a soil sample. If another agent believes that a soil sample can also be found at w1 and no other agent contradicts this information, then then agent will not need to move to w2, thus saving one planning action. In these cases, the plans contain fewer actions because agents beliefs are also used to support the fulfilment of an open goal. On the other hand, the difference in the duration of MAP-POP plans with respect to Q-DeLP-MAP and PS-Q-DeLP-MAP plans is rather noticeable because it is usually the case that the fewer actions in a plan, the fewer time steps.

Additionally, we wanted to assess the feasibility of the solution plans of MAP-POP. Given a context, we say that a solution plan is feasible if it does not contain actions that would be otherwise discarded in an argumentation process. That is, feasibility is a rough measure to know if a solution would contain actions that an argumentation process would label as failing actions, as a result of the argumentative dialogues among the agents (defeated arguments) and according to the context information. In this sense, we analyzed the plans of MAP-POP that were actually discarded by the other two approaches since some of the actions of the plans were labeled as failing actions. For instance, in the rovers domain, at least 50% of the actions of the solution plan of Problem3 were acknowledged not to be successfully executable in Q-DeLP-MAP and PS-Q-DeLP-MAP. Similarly, for Problem5 in logistics, 25% of the actions in the solution plan of MAP-POP were cataloged as failing action according to Q-DeLP-MAP and PS-Q-DeLP-MAP.

	P1	P2	P3	P4	P5
Rover 0	100,0%	100,0%	25,0%	29,2%	16,7%
Rover 1		0,0%	75,0%	70,8%	44,4%
Rover 2					38,9%

(a) Rovers.

	P1	P2	P3	P4	P5
Truck 1	25,0%	28,0%	13,5%	32,3%	44,4%
Truck 2	50,0%	32,0%	24,3%	16,1%	8,3%
Truck 3			18,9%	16,1%	16,6%
Airplane 1	25,0%	40,0%	43,2%	35,5%	30,5%

(b) Logistics.

Figure 2.55: Evaluating the agents' contribution level in the solution plans (with high contradiction) of rovers and logistics in Q-DeLP-MAP.

Finally, we were also interested in checking the contribution level of the agents in the solution plans of Q-DeLP-MAP. This is calculated as the number of action-arguments or supporting arguments contributed by each agent to the plans. Figure 2.55 shows the contribution of each agent to the the solution plan of each problem. For instance, Figure 2.55(a) shows that in Problem5 the three rovers collaboratively participate in the construction of the solution. The same is observed in the logistics results of Figure 2.55(b).

2.7.6 Conclusions and Future Work

This paper describes Q-DeLP-MAP, a domain-independent Argumentation-based Multi-Agent Planning system. The main contributions of this paper are: (i) the extension of DeLP-POP framework for Multi-Agent Planning and for the *qualification problem*; (ii) the presentation of all the technical components of Q-DeLP-MAP; (iii) the experimental evaluation with two domains from the IPC and comparison to two other MAP approaches.

We conclude that using argumentation is a promising line to tackle planning problems while incorporating the agents beliefs within the reasoning process. This allows us to consider unexpected environmental conditions which cannot be modeled within the planning representation. Additionally, the extension of the model to a multi-agent system opens many possibilities of application in real-world problems where knowledge, abilities and beliefs are distributed across several entities.

3

General discussion of the results

The objective of this chapter is to summarize the main contributions of this PhD thesis presented in Chapter 2. This chapter is organized as follows: section 3.1 presents the contributions related to the theoretical model; section 3.2 summarizes the main features of the planning language; section 3.3 shows the functional components of the framework; and, finally, section 3.4 presents a summary of the experimental results.

3.1 Theoretical Model

One of the first steps of this research work was to analyze the key components of a MAP task in order to later integrate it with the argumentation model. Our definition of a MAP task draws upon the specification of a cooperative MAP task presented in MAP-POP (33). This definition was progressively updated and revised to include the defeasible knowledge of the agents.

The specification of a MAP task includes a non-empty set of agents $\mathsf{AG} = \{\mathsf{Ag}_1 \ldots \mathsf{Ag}_n\}$ with planning and argumentation capabilities. Each agent Ag_i is endowed with a tuple $\mathbb{M}_{Ag_i} = \langle \Psi_{\mathsf{Ag}_i}, \Delta_{\mathsf{Ag}_i}, A_{\mathsf{Ag}_i}, G \rangle$, where: Ψ_{Ag_i} represents the partial view of the initial state of agent Ag_i; Δ_{Ag_i} is the set of defeasible rules known by agent Ag_i; A_{Ag_i} is the set of planning actions known by agent Ag_i; and, G is a set of global goals that represent the needs of a user in an environment. Since we are dealing with cooperative planning, the goal state G is known to all agents as they will all contribute to the solution that achieves

G.

The agents in AG jointly solve G and help discover the non-anticipating conditions that might prevent the actions in A from being executable in their particular context of application. All details of the MAP task have been explained throughout the compendium of articles of this PhD thesis: sections 2.3.4.1, 2.5.2.3, 2.6.4.3, 2.7.4.1.

In order to allow agents to argue about the executability of an action of the plan, every action is encoded in the form of an argument so that any agent that has information about a tentative anomalous situation that might prevent the action from being executed launches an attack against the argument representing the action. Q-DeLP-MAP represents an action of the plan as an action-argument, which consists of a fictitious argument (that derives the real effects of the action) and an action that now derives a fictitious effect and supports the base of the fictitious argument. More specifically, a planning action α is internally translated into a pair $\langle \alpha', \mathcal{A} \rangle$ where α' is used to denote the execution of α and the argument $\mathcal{A}$ embodies the achievement of the effects of α. This way, an attack to $\mathcal{A}$ is an attack to the successful execution of α. All the details are widely explained in section 2.7.3.1.

The action-argument representational scheme is the baseline of our framework to address the formalization of the *qualification problem*. In planning, the default choice to the qualification problem is to assume away the numerous possible unexpected circumstances that may prevent an action from being executed in the real world and resort to replan in case a plan turns out to not being executable due to a non-anticipating condition. The theoretical argumentation MAP model presented in this thesis is precisely a method to address the qualification problem by integrating the local know-how knowledge of the agents within the planning task. Agents put forward an action α to the plan in the form of a action-argument $\langle \alpha', \mathcal{A} \rangle$ and any other agent that anticipates an unexpected condition that affects the application of α generates an attacking argument $\mathcal{B}$ against $\mathcal{A}$. In turn, an agent whose beliefs contradict $\mathcal{B}$, activates an attacking argument $\mathcal{C}$ against $\mathcal{B}$ and so on. This way, in Q-DeLP-MAP, all the components of a partial plan are treated as defeasible knowledge and agents' beliefs as attacking arguments against the action-arguments of the plan. Details on the attacking arguments can be seen in section 2.7.3.1.

Another contribution of the theoretical model is a comprehensive classification of the conflicting situations in threats and attacks. We propose a similar representation scheme for both types of interferences. In the case of threats, we provide a thorough analysis of

the threats that arise in a plan according to the type of link that is being threatened. As for the attacks, we represent the argument of the plan under attack and the external attacking arguments. Threats and attacks share a similar semantics: a threat represents a conflicting situation according to the causal theory that governs the physical world whereas an attack represents a conflicting situation according to the know-how knowledge of the agents. This unified view of conflicts is one of the distinguishable features of Q-DeLP-MAP. Details on the conflicting situations can be seen in section 2.7.3.2.

Finally, we provide a detailed formalization of the concept of *proto-state* and its role in the activation of the attacking arguments. Even though this notion of propagated effects was previously introduced in other approaches, we provide a more comprehensive view of the *proto-state* of an action-argument thanks to the use of a uniform representational scheme for action-arguments, threats and attacks. Details on the *proto-states* can be seen in section 2.7.3.3.

3.2 Language of Planning and Argumentation

Our planning language is based on PDDL3.1, the latest version of PDDL (Planning Domain Definition Language). Unlike its predecessors, that model a planning domain through logical predicates, PDDL3.1 also incorporates state variables by adding object fluents that map a tuple of objects to an object of the problem. More specifically, we assume the PDDL codification of the multi-agent planning tasks presented by MAP-POP (33). This requires the specification of multiple planning problems, one per agent, defining the abilities, initial state and planning context of each agent. Since information of the planning task is distributed across agents, we also create specific structures to define the information that agents will exchange between each other during the planning process. Additionally, we extended the PDDL3.1 language presented by MAP-POP with the set of defeasible rules of the agents defined through the additional constructor `:def-rule`.

Listing 3.1 shows an example of a planning action from the rovers domain. The section `:precondition` is the same as in a classical PDDL action. The section `:effects` contains the effects of the action as specified in the domain, and the fictitious effect (*(communicated-established ?p ?x ?y)*). Q-DeLP-MAP converts this action into a pair $\langle \alpha, \mathcal{A} \rangle$ where $\mathsf{concl}(\mathcal{A})$ = *(communicated-soil-data ?p)*, that is, the real effects of the action. Thus, our system automatically transforms the action into an action-argument,

resulting in an action α' that support the basis *(communicated-established ?p ?x ?y)* of the fictitious argument $\mathcal{A}$ which in turn derives the real effects *(communicated-soil-data ?p)*. In a first version of the language (section 2.6.5.1), the user was required to explicitly encode a planning action as an action and a defeasible rule. However, in Q-DeLP-MAP, the system performs this automatically so it is transparent to the user.

```
(:action communicate-soil-data
 :parameters (?r - rover ?l - lander ?p - waypoint ?x - waypoint ?y - waypoint)
 :precondition (and
   (myRover ?r)
   (= (at ?r) ?x)
   (= (at_lander ?l) ?y)
   (have_soil_analysis ?r ?p)
   (visible ?x ?y))
 :effect (and
   (communicated-soil-data ?p)
   (communicated-established ?p ?x ?y)))
```

Listing 3.1: Example of an action for communicating the soil data of a rover.

Listing 3.2 and 3.3 show two defeasible rules that represent the beliefs of an agent or of two distinct agents. In the first rule, assuming the agent is aware of a solar storm through some available source of information, the agent will infer communication problems in the particular waypoint. On the basis of the existence of communication problems, in the second rule, the agent will deduce that the soil data can not be communicated properly.

```
(:def-rule R-communication-problems
   :parameters(?x - waypoint ?y - waypoint)
   :body (solar-storm ?x)
   :head (communication-problems ?x ?y))
```

Listing 3.2: Rule 1: Example of a defeasible rule of communication problems.

```
(:def-rule R-communicate_soil_data-Denied
   :parameters(?r - rover ?l - lander ?p - waypoint ?x - waypoint ?y - waypoint)
   :body (and
     (communication-problems ?x ?y)
     (communicated-established ?p ?x ?y))
   :head (not (communicated-soil-data ?p)))
```

Listing 3.3: Rule 2: Example of a defeasible rule of communication problems.

Given these two defeasible rules, an agent can build an argument against the fictitious argument that is generated by the action *communicate-soil-data*, provided that there are solar storms.

Finally, the language is designed to encode MAP tasks in a factored way: each agent receives a separate domain and problem that model its knowledge of the MAP task.

3.3 Framework for cooperative distributed planning

One of the main results of this PhD thesis is to come up with an operative, domain-independent and fully-integrated argumentation and planning framework for multi-agent contexts. The functional architecture of Q-DeLP-MAP follows the same stages of a traditional partial-order planner but it features some distinctive characteristics due to the existence of multiple agents planning and arguing together. The starting point is an empty plan and agents start a stepwise dialogue consisting of plan proposals plus arguments against them. More specifically:

Plan Generation. Plan refinements are the plan proposals put forward by the agents as a result of expanding a selected plan when solving an open goal. This stage follows a process similarly to a plan-space planning process that builds a POP tree, except that each refinement or successor of the selected plan can be now generated by a different agent and can contain arguments to support the action preconditions or argument base. Agents exchange their refinements to each other and learn the new steps, literals and links of the plan, which they keep in their local POP tree. In Q-DeLP-MAP, the coordination of the plan proposals is carried out by a coordination protocol based on a *democratic leadership* where a leadership baton is scheduled among the agents following a round-robin strategy. Once the baton agent has sent his refinements to the rest of the agents, and acknowledgment is received, the coordination stage is completed, and the baton is handed over to the following agent. The process is repeated until all the agents have once taken the baton role.

Plan Evaluation. Evaluating a plan and analyzing its executability in the real world, according to the agents' beliefs, is done through two alternative protocols: Plan Argumentation and Case-Based Reasoning (CBR) (62, 118). In the Plan Argumentation stage, agents become engaged in a series of argumentative dialogues aimed at evaluating the guarantee of a successful execution of a plan proposal. Specifically, argumentation oc-

curs in the context of the agents' beliefs and they build arguments to defend or attack a plan proposal. Subsequently, the CBR will register the argumentative case in order to be able to re-use it in further evaluations. Before starting a dialectical tree over an argument $\mathcal{A}$ that supports an action-argument γ under a proto-state s_γ, we search for the existence of an identical case in the database, in which case the argument is directly labeled with the same evaluation result (defeated or undefeated). Otherwise, a new dialectical tree is launched.

The first prototype of the multi-agent planning and argumentation architecture is described in paper (34), where we introduce the main components of the model and their relationships. Subsequently, in papers (26), (27) and (28), we formally present the multi-agent argumentative dialogues. Finally, paper (35) provides a complete and detailed overview of the cooperative planning protocol, highlighting the functional behavior of Q-DeLP-MAP.

3.4 Empirical Evaluation

In this work, we present two sets of experiments to assess the performance and scalability of the model. The first set of experiments evaluate the model in a particular application of ambient intelligence in the field of health-care (27) and (28). In the second test of experiments, we test Q-DeLP-MAP in two well-known domains of the IPC benchmarks: the rovers domain and the logistics domain. With this whole experimentation, we accomplish our objective of designing and building a domain-independent framework.

Regarding the evaluation of the model in the health-care application, we created several planning problems with three types of agents (transport, communication and assistant) and we measured the performance of our model (CAMAP) compared to a traditional MAP system with no argumentation. Results corroborate that the computational time considerably increases when argumentation is included in the reasoning process. However, we observe an improvement in the quality of the solutions according to the number of actions and makespan of the solution plans. Interestingly, many solutions returned by CAMAP comprised supporting arguments, meaning that the beliefs of the agents were also used to support open goals of the plan without need to specifically include a planning action. That is, the know-how knowledge and expertise of the agents, which is not encoded in the form of planning actions, is also a very valuable source of information to

achieve goals of the problem. The results also show that, for small problems, the solutions returned by the non-argumentative model included at least 30% of the actions that CAMAP agents acknowledged not to be successfully executed.

In section 2.7.5, we evaluated Q-DeLP-MAP with several problems from the rovers and logistics domain from the IPC benchmarks. The logistics problems fall into the tightly-coupled category since agents (trucks and planes) have to cooperate to transport the different packages to the target locations and problems present several coordination points (locations) at which agents can interact. However, the problems from the rovers domain present a medium coupling level: rover agents are independent but they have access to certain shared resources in their environment, namely the rock and soil samples they collect and analyze.

The details and figures of the experimentation are shown in section 2.7.5. The more complex problem Q-DeLP-MAP was able to solve was an instance of the logistics domain comprising: 174 actions, 164 defeasible rules, 9 goals, 4 agents and a high level of contradiction between the information of the agents. Q-DeLP-MAP took 480 seconds to solve this problem whereas MAP-POP (with no argumentation) took 300 seconds. Nevertheless, in average, Q-DeLP-MAP computation time is only slightly higher than MAP-POP.

Although the performance worsens with the use of argumentation, as it also happened in the health-care application, we think it is a minimum cost overrun. The reason is that in Q-DeLP-MAP agents need to exchange thousands of messages in order to guarantee the validity and feasibility of the plan. At this point, it is worth noting that the use of the CBR module brought a significant improvement in the overall performance.

The results in section 2.7.5 also show that the solution plans of Q-DeLP-MAP are more robust than those of MAP-POP since actions have a higher guarantee of a successful execution (are not likely to fail at execution time). This is widely discussed in sections 2.5.4, 2.6.7 and 2.7.5. More specifically, in general, at least 50% of the actions in the plans returned by MAP-POP were labeled as defeated in Q-DeLP-MAP. This confirms that the beliefs of the agents about the environment can be successfully used during the construction of a plan to assess the contextual conditions that affect the applicability of the actions and which are not encoded in the causal theory of the planning task.

4

Conclusions

This PhD contributes with an argumentation-based model for solving MAP problems that incorporates the know-how knowledge of the agents in the form of defeasible knowledge to determine the feasibility of the actions when executed in the real world. The contents of this document present the chronological achievements and successive refinements of the argumentation model to eventually come up with Q-DeLP-MAP, the final version of the model, which provides a unified representational framework for planning and argumentation and notably improves the performance of the overall system.

We highlight two main strengths of Q-DeLP-MAP:

(1) To the best of our knowledge, Q-DeLP-MAP is the first attempt to integrate an argumentation-based model within a multi-agent planner. Unlike most argumentation MAP models, aimed at solving conflicts that arise among the plans calculated by each individual agent with a single-agent state-of-the-art planner, in our proposal the argumentation model is embedded into MAP-POP. This way, planning and argumentation continuously feed each other during the process of building a plan and the arguments of the agents are used to prune those plans that turn out to be defeated, as a probably indication of a failing execution. In the experiments presented in the paper (35), we show the benefits of interleaving planning and argumentation versus using a one-shot planning and argumentation procedure.

(2) Q-DeLP-MAP is a domain-independent model that can be applied to any planning

application and provides an easy-to-use, PDDL-like language for defining defeasible rules. Defeasible knowledge is expressed with the same state variables and argument terms used for the planning task specification and the internal conversion of actions to action-argument steps is totally transparent to the user. This unified representational scheme allows Q-DeLP-MAP to handle all the planning and argumentation components uniformly, which results in an efficient management of the internal structures. Q-DeLP-MAP has been extensively tested on a practical healthcare application as well as on some of the IPC benchmarks, which demonstrates its versatility, flexibility and adaptability to be used in any application context.

As for weaknesses of the approach, we enumerate some of the current limitations of Q-DeLP-MAP and possible ways to overcome them in future developments of the model:

(1) Even though all the achievements done in the model greatly improved Q-DeLP-MAP performance, including the incorporation of a CBR module, the solving capability of Q-DeLP-MAP is still limited as it is unable to solve problems that involve a high number of agents, actions and defeasible rules. One matter we did not deal with in this PhD dissertation is the heuristic functions. We simply adopted the classical POP heuristics, which are known not to be very informative. In this regard, there are two possible ways of action:

   (a) Incorporate state-based heuristics in Q-DeLP-MAP. We must note that state-based heuristics applied to POP imply to change the classical POP search scheme so as to generate problem states in which the heuristic can be applied. This is the usual approach adopted by forward-chaining partial-order planners. However, since Q-DeLP-MAP internally computes the *proto-states* of the actions, we could make use of this concept to regressively calculate a state-based heuristic from the *proto-state* of an action-argument to the initial state, instead of from the problem state to the goal state. This way, we could benefit of the more informative state-based heuristics without need to change the backwards Q-DeLP-MAP search scheme.

   (b) The heuristic function of Q-DeLP-MAP is exclusively calculated in terms of planning, no argumentation items are involved in its calculation, and it only prioritizes the selection of undefeated plans. Our hypothesis is that the search guidance could be improved by incorporating argumentation information in

the heuristic function; for instance, a prediction of the potential number of attacks that a refinement plan might receive in the subsequent iterations.

(2) Defeasible knowledge not only changes in the light of new evidences brought by the agents of the system but also as a result of a changing world that impacts the beliefs of the agents. Q-DeLP-MAP is an offline system and defeasible rules are static so the issue of introducing new information that may alter the beliefs of the agents is not addressed in this work. In order to incorporate this desirable property and make Q-DeLP-MAP a truly adaptive system, a belief revision mechanism that updates the status of the defeated and undefeated arguments of the plans should be included. The use of the CBR module is crucial here since argumentative dialogues would be revised only once instead of checking every appearance of the argument in the partial plans of the search tree. On the other hand, this belief revision would not imply any change in the planning process, just revising the nodes of the tree and updating the lists of nodes accordingly.

(3) Defeasible knowledge of the agents basically stems from the expertise of the agents and the information sources available to them. In Q-DeLP-MAP, all agents' beliefs are given the same importance and no reliability or accuracy measure is used to distinguish or rank them. Thus, a key improvement would be to associate beliefs to a degree of trust according to the origin of the belief; i.e., reliability of the information source, reputation of the agent, etc. A trust level associated to the beliefs would be used to classify the attacking arguments so that agents would be only allowed to attack arguments with similar or lower trust level. Likewise, old evidences would prevail over new evidences if they hold a greater trust level.

All in all, Q-DeLP-MAP is an easily extensible and adaptable model thanks to the modular design of the framework, the use of domain-independent, off-the-shelf planning technology and a unified representational scheme for planning and argumentation.

References

[1] JOHN MCCARTHY AND PATRICK J HAYES. **Some philosophical problems from the standpoint of artificial intelligence**. *Readings in artificial intelligence*, pages 431–450, 1969. 2, 155

[2] R. GIRLE, D. L. HITCHCOCK, P. MCBURNEY, AND B. VERHEIJ. **Decision support for practical reasoning: A theoretical and computational perspective**. In *C. Reed and T.J. Norman, eds., Argumentation Machines. New Frontiers in Argument and Computation*, pages 55–84, 2003. 2, 155

[3] J. POLLOCK. **Defeasible Planning**. In *AAAI Conference on Artificial Intelligence (AAAI) Workshop, Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, Carnegie Mellon University, June*, **7**, 1998. 2, 155

[4] PHAN MINH DUNG. **On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games**. *Artificial Intelligence Journal*, **77**(2):321–358, 1995. 2, 49, 92, 156

[5] IYAD RAHWAN AND LEILA AMGOUD. **An Argumentation-based Approach for Practical Reasoning**. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 347–354, 2006. 2, 23, 48, 49, 71, 92, 156

[6] JORIS HULSTIJN AND LEENDERT WN VAN DER TORRE. **Combining goal generation and planning in an argumentation framework.** In *International Workshop on Nonmonotonic Reasoning (NMR), in conjunction with Principles of Knowledge Representation and Reasoning (KR)*, pages 212–218, 2004. 2, 92, 156

[7] LEILA AMGOUD AND SOUHILA KACI. **An argumentation framework for merging conflicting knowledge bases**. *International Journal of Approximate Reasoning*, **45**(2):321–340, 2007. 3, 156

[8] ARIEL MONTESERIN AND ANALÍA AMANDI. **Argumentation-based negotiation planning for autonomous agents**. *Decision Support Systems (DSS) Journal*, **51**(3):532–548, 2011. 3, 156

[9] D.N. WALTON. *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates, Mahwah, NJ, 1996. 3, 156

[10] KATIE ATKINSON AND TREVOR J. M. BENCH-CAPON. **Practical reasoning as presumptive argumentation using action based alternating transition systems**. *Artificial Intelligence Journal*, **171**(10-15):855–874, 2007. 3, 156

[11] PANCHO TOLCHINSKY, SANJAY MODGIL, AND ULISES CORTÉS. **Argument Schemes and Critical Questions for Heterogeneous Agents to Argue over the Viability of a Human Organ for Transplantation.** In *AAAI Spring Symposium: Argumentation for Consumers of Healthcare*, **6**, pages 105–105, 2006. 3

[12] TREVOR J. M. BENCH-CAPON. **Persuasion in Practical Argument Using Value-based Argumentation Frameworks**. *Journal of Logic and Computation*, **13**(3):429–448, 2003. 3

[13] KATIE ATKINSON AND TREVOR J. M. BENCH-CAPON. **Legal Case-based Reasoning as Practical Reasoning**. *Artificial Intelligence Law*, **13**(1), 2005. 3, 156

[14] DAN CARTWRIGHT AND KATIE ATKINSON. **Using Computational Argumentation to Support E-participation**. *IEEE Intelligent Systems*, **24**(5):42–52, 2009. 3, 156

[15] TREVOR J. M. BENCH-CAPON, KATIE ATKINSON, AND PETER MCBURNEY. **Using argumentation to model agent decision making in economic experiments**. *Autonomous Agents and Multi-Agent Systems*, **25**(1):183–208, 2012. 3, 156

[16] ROLANDO MEDELLIN-GASQUE, KATIE ATKINSON, TREVOR J. M. BENCH-CAPON, AND PETER MCBURNEY. **Strategies for question selection in argumentative dialogues about plans**. *Argument & Computation*, **4**(2):151–179, 2013. 4, 6, 156, 157

[17] E. KOK, J.J. MEYER, H. PRAKKEN, AND G. VREESWIJK. **Testing the Benfits of Structured Argumentation in Multi-Agent Deliberation Dialogues**. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1411–1412, 2012. 4, 156

[18] O. SAPENA, A. TORREÑO, AND E. ONAINDÍA. **On the Construction of Joint Plans through Argumentation Schemes**. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1195–1196, 2011. 4, 50, 156

[19] JOHN L. POLLOCK. **How to Reason Defeasibly**. *Artificial Intelligence Journal*, **57**(1):1–42, 1992. 4, 156

[20] JOHN L. POLLOCK. **Rational Cognition in OSCAR**. In *nternational Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL)*, pages 71–90, 1999. 4, 156

[21] GEORGE FERGUSON AND JAMES F. ALLEN. **Arguing about Plans: Plan Representation and Reasoning for Mixed-initiative Planning**. In *Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 43–48, 1994. 4, 156

[22] GUILLERMO R. SIMARI, ALEJANDRO J. GARCÍA, AND MARCELA CAPOBIANCO. **Actions, planning and defeasible reasoning**. In *International Workshop on Nonmonotonic Reasoning (NMR), in conjunction with Principles of Knowledge Representation and Reasoning (KR)*, pages 377–384, 2004. 5, 6, 23, 156

[23] ALEJANDRO J. GARCÍA DIEGO R. GARCÍA AND GUILLERMO R. SIMARI. **Defeasible reasoning and partial order planning**. In *International Conference on Foundations of Information and Knowledge Systems*, pages 311–328. Springer-Verlag, 2008. 5, 18, 19, 23, 29, 30, 45, 48, 50, 51, 54, 58, 62, 72, 74, 75, 76, 77, 79, 85, 97, 100, 101, 119, 123, 126, 157, 159, 163, 164, 168, 170

[24] ALEJANDRO J. GARCÍA AND GUILLERMO R. SIMARI. **Defeasible Logic Programming: An Argumentative Approach**. *Theory and Practice of Logic Programming*, **4**(2):95–138, 2004. 5, 6, 7, 10, 23, 25, 26, 36, 48, 71, 73, 74, 75, 95, 108, 122, 142, 143, 157, 159, 161, 162, 182

[25] DIEGO R GARCIA, ALEJANDRO J GARCIA, AND GUILLERMO R SIMARI. **Planning and defeasible reasoning**. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, page 222, 2007. 5

[26] P. PARDO, S. PAJARES, EVA ONAINDÍA, L. GODO, AND P. DELLUNDE. **Multiagent Argumentation for Cooperative Planning in DeLP-POP**. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 971–978, 2011. 5, 20, 21, 48, 49, 51, 56, 57, 73, 79, 97, 101, 121, 123, 124, 126, 157, 158, 160, 175, 180, 200

[27] S. PAJARES AND E. ONAINDÍA. **Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications**. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 509–516, 2012. 5, 7, 21, 93, 158, 165, 173, 183, 200

[28] S. PAJARES FERRANDO AND E. ONAINDÍA. **Context-Aware Multi-Agent Planning in intelligent environments**. *Information Sciences Journal*, **227**:22–42, 2013. 5, 21, 158, 165, 183, 185, 186, 200

[29] A. BELESIOTIS, M. ROVATSOS, AND I. RAHWAN. **Agreeing on Plans Through Iterated Disputes**. In *Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 765–772, 2010. 5, 6, 50, 92, 120, 141, 157, 184, 189

[30] A. BELESIOTIS. *Argumentation-Based Methods for Multi-Perspective Cooperative Planning*. PhD thesis, University of Edinburgh, United Kingdom, 2012. 5, 6, 157, 184, 189

[31] A. TONIOLO, T. NORMAN, AND K. SYCARA. **Argumentation Schemes for Collaborative Planning**. *Agents in Principle, Agents in Practice*, pages 323–335, 2011. 6, 120, 141, 157

[32] A. TONIOLO, T. NORMAN, AND K. SYCARA. **On the benefits of argumentation schemes in deliberative dialogue**. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1409–1410, 2012. 6, 157

[33] A. TORREÑO, E. ONAINDÍA, AND O. SAPENA. **An approach to MultiAgent Planning with Incomplete Information**. In *Conference on Artificial Intelligence (ECAI)*, **242**, pages 762–767. IOS Press, 2012. 6, 7, 10, 121, 183, 195, 197

[34] S. PAJARES FERRANDO, E. ONAINDÍA, AND A. TORREÑO. **An Architecture for Defeasible-Reasoning-Based Cooperative Distributed Planning**. In *International Conference on Cooperative Information Systems (CoopIS), in conjunction with On the Move to Meaningful Internet Systems (OTM)*, pages 200–217, 2011. 7, 93, 97, 121, 123, 124, 126, 131, 158, 165, 200

[35] S. PAJARES FERRANDO AND E. ONAINDÍA. **Argumentation-based Planning**. *Submitted to Artificial Intelligence Journal (AIJ)*, 2015. 7, 200, 203

[36] MALIK GHALLAB, DANA NAU, AND PAOLO TRAVERSO. *Automated planning: theory & practice*. 2004. 22, 33, 118

[37] T. J. M. BENCH-CAPON AND P. E. DUNNE. **Argumentation in artificial intelligence**. *Artificial Intelligence Journal*, **171**(10-15):619–641, 2007. 22, 48, 71

[38] HENRY PRAKKEN AND GIOVANNI SARTOR. **Argument-Based Extended Logic Programming with Defeasible Priorities**. *Journal of Applied Non-Classical Logics*, **7**(1), 1997. 23

[39] H. PRAKKEN, T. GORDON, D. WALTON, T. BENCH-CAPON, FJ BEX, SW VAN DEN BRAAK, H. VAN OOSTENDORP, H. PRAKKEN, HB VERHEIJ, AND GAW VREESWIJK. **Logical Tools for Modelling Legal Argument: A Study of Defeasible Reasoning in Law**. *Dordrecht, Boston*, 1997. 23, 71

[40] MATTHIAS THIMM AND GABRIELE KERN-ISBERNER. **A Distributed Argumentation Framework using Defeasible Logic Programming**. In *International Conference on Computational Models of Argument (COMMA)*, pages 381–392, 2008. 23

[41] M. THIMM. **Realizing Argumentation in Multi-agent Systems Using Defeasible Logic Programming**. In *International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), in conjunction with International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 175–194, 2009. 23, 50, 92

[42] A. BARRETT AND D.S. WELD. **Partial-order planning: evaluating possible efficiency gains**. *Artificial Intelligence Journal*, **67**(1):71–112, 1994. 23, 27

[43] J.S. PENBERTHY AND D. WELD. **UCPOP: A sound, complete, partial order planner for ADL**. In *International Conference on Knowledge Representation and Reasoning (KR)*, pages 103–114, 1992. 23, 27, 48, 71, 96, 119, 123, 162

[44] CARLOS I. CHESÑEVAR FRIEDER STOLZENBURG, ALEJANDRO J. GARCÍA AND GUILLERMO R. SIMARI. **Computing generalized specificity**. *Journal of Applied Non-Classical Logics*, **13**(1):87, 2003. 25, 37

[45] R.E. FIKES AND N.J. NILSSON. **STRIPS: A new approach to the application of theorem proving to problem solving**. *Artificial Intelligence Journal*, **2**(3-4):189–208, 1971. 26, 65

[46] DANIEL S WELD. **An introduction to least commitment planning**. *AI magazine*, **15**(4):27, 1994. 27, 162, 172

[47] JÖRG HOFFMANN AND RONEN I. BRAFMAN. **Conformant planning via heuristic forward search: A new approach**. *Artificial Intelligence Journal*, **170**(6-7):507–541, 2006. 45

[48] M. BRENNER AND B. NEBEL. **Continual Planning and Acting in Dynamic Multiagent Environments**. *Journal of Autonomous Agents and Multiagent Systems*, **19**(3):297–331, 2009. 45

[49] M. MECELLA, M. SCANNAPIECO, A. VIRGILLITO, R. BALDONI, T. CATARCI, AND C. BATINI. **Managing data quality in cooperative information systems**. In *International Conference on Cooperative Information Systems (CoopIS), in conjunction with On the Move to Meaningful Internet Systems (OTM)*, pages 486–502, 2002. 47

[50] M. WOOLDRIDGE. **Agent-based software engineering**. In *Software Engineering. IEE Proceedings*, **144**, pages 26–37. IET, 1997. 47

[51] L. AMGOUD. **A formal framework for handling conflicting desires**. In *7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 552–563. Springer, 2004. 49, 92

[52] J. POLLOCK. **Defeasible planning**. In *AAAI Conference on Artificial Intelligence (AAAI) Workshop, Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, Carnegie Mellon University, June*, 1998. 49, 50

[53] Y. TANG, T. NORMAN, AND S. PARSONS. **A model for integrating dialogue and the execution of joint plans**. In *International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), in conjunction with International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 60–78, 2010. 50, 92

[54] D.E. SMITH, J. FRANK, AND A.K. JÓNSSON. **Bridging the Gap Between Planning and Scheduling**. *The Knowledge Engineering Review*, **15**(1):47–83, 2000. 51, 71, 96, 123

[55] M. GHALLAB AND H. LARUELLE. **Representation and Control in IxTeT, a Temporal Planner**. In *Conference on Artificial Intelligence Planning Systems (AIPS)*, **94**, pages 61–67, 1994. 51, 71

[56] A. JONSSON, P. MORRIS, N. MUSCETTOLA, K. RAJAN, AND B. SMITH. **Planning in Interplanetary Space: Theory and Practice**. In *International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pages 177–86, 2000. 51, 71, 96

[57] P. HASLUM AND H. GEFFNER. **Admissible Heuristics for Optimal Planning**. In *Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 140–149, 2000. 51

[58] A.L. BLUM AND M.L. FURST. **Fast Planning Through Planning Graph Analysis**. *Artificial intelligence Journal*, **90**(1-2):281–300, 1997. 51

[59] MATTHEW L. GINSBERG AND DAVID E. SMITH. **Reasoning About Action II: The Qualification Problem**. *Artificial Intelligence Journal*, **35**(3):311–342, 1988. 54, 77, 101, 128, 164, 165

[60] A. GEREVINI AND L. SCHUBERT. **Accelerating partial-order planners: Some techniques for effective search control and pruning**. *Journal of Artificial Intelligence Research*, **5**:95–137, 1996. 58, 85

[61] C.J. IBBOTT AND R.M. O'KEEFE. **Trust, planning and benefits in a global interorganizational system**. *Information Systems Journal*, **14**(2):131–152, 2004. 62

[62] AGNAR AAMODT AND ENRIC PLAZA. **Case-based reasoning: Foundational issues, methodological variations, and system approaches**. *Artificial Intelligence (AI) Communications*, **7**(1):39–59, 1994. 62, 182, 199

[63] A. BIKAKIS AND G. ANTONIOU. **Distributed Defeasible Contextual Reasoning in Ambient Computing**. *Ambient Intelligence*, pages 308–325, 2008. 64, 95

[64] E. KONTOPOULOS, N. BASSILIADES, AND G. ANTONIOU. **Visualizing Semantic Web proofs of defeasible logic in the DR-DEVICE system**. *Knowledge-Based Systems Journal*, **24**(3):406–419, 2011. 64

[65] S. PAJARES AND E. ONAINDÍA. **Temporal Defeasible Argumentation in Multi-Agent Planning**. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2834–2835, 2011. 70, 93, 114

[66] GUILLERMO R. SIMARI AND RONALD P. LOUI. **A mathematical treatment of defeasible reasoning and its implementation**. *Artificial Intelligence Journal*, **53**(2-3):125–157, 1992. 74, 159, 161

[67] JUDEA PEARL. **Heuristics: intelligent search strategies for computer problem solving**. 1984. 85

[68] R. MEDELLIN-GASQUE, K. ATKINSON, P. MCBURNEY, AND T. BENCH-CAPON. **Arguments over co-operative plans**. In *International Workshop on Theory and Applications of Formal Argumentation (TAFA)*, 2011. 92

[69] R. MEDELLIN-GASQUE, K. ATKINSON, P. MCBURNEY, AND T. BENCH-CAPON. **Critical Questions for Plan Proposals**. Technical report, Technical Report ULCS-11-003, Department of Computer Science, University of Liverpool, UK, 2011. 92

[70] IYAD RAHWAN, SARVAPALI D RAMCHURN, NICHOLAS R JENNINGS, PETER MCBURNEY, SIMON PARSONS, AND LIZ SONENBERG. **Argumentation-based negotiation**. *The Knowledge Engineering Review*, **18**(04):343–375, 2003. 93

[71] PERE PARDO, PILAR DELLUNDE, AND LLUÍS GODO. **Argumentation-based Negotiation in t-DeLP-POP.** In *Conference of the Catalan Association for Artificial Intelligence (CCIA)*, pages 179–188, 2011. 93

[72] PERE PARDO AND LLUÍS GODO. **t-DeLP: a temporal extension of the defeasible logic programming argumentative framework**. In *Scalable Uncertainty Management*, pages 489–503. Springer, 2011. 93

[73] S. PAJARES AND E. ONAINDÍA. **Defeasible Planning through Multi-Agent Argumentation**. *Modelling Machine Emotions For Realizing Intelligence, Smart Innovation Systems and Technologies Series*, **1**:1–19, 2011. 93, 97, 121, 123, 124, 165

[74] EMILE AARTS. **Ambient intelligence**. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 1–1. Springer, 2004. 94, 117

[75] H. PRAKKEN AND G. VREESWIJK. **Logics for defeasible argumentation**. *Handbook of philosophical logic*, **4**:219–318, 2002. 95, 122

[76] ANTONIS BIKAKIS AND GRIGORIS ANTONIOU. **Defeasible contextual reasoning with arguments in ambient intelligence**. *Knowledge and Data Engineering, IEEE Transactions on*, **22**(11):1492–1506, 2010. 95, 117, 119

[77] F. AMIGONI, N. GATTI, C. PINCIROLI, AND M. ROVERI. **What Planner for Ambient Intelligence Applications?** *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, **35**(1):7–21, 2005. 96, 104, 118, 119

[78] DANIEL L KOVACS. **Complete BNF description of PDDL3**. 2011. 102, 132

[79] X.L. NGUYEN AND S. KAMBHAMPATI. **Reviving Partial Order Planning**. In *International Joint Conference on Artificial Intelligence*, **17**, pages 459–466, 2001. 110, 145, 175, 183

[80] MATTHIAS BALDAUF, SCHAHRAM DUSTDAR, AND FLORIAN ROSENBERG. **A survey on context-aware systems**. *International Journal of Ad Hoc and Ubiquitous Computing*, **2**(4):263–277, 2007. 116

[81] JONG-YI HONG, EUI-HO SUH, AND SUNG-JIN KIM. **Context-aware systems: A literature review and classification**. *Expert Systems with Applications*, **36**(4):8509–8522, 2009. 116

[82] ANIND K DEY. **Understanding and using context**. *Personal and ubiquitous computing*, **5**(1):4–7, 2001. 116

[83] ALBRECHT SCHMIDT. *Ubiquitous computing-computing in context*. PhD thesis, Lancaster University, 2003. 116

[84] EMILE HL AARTS AND BORIS ER DE RUYTER. **New research perspectives on Ambient Intelligence.** *Journal of Ambient Intelligence and Smart Environments (JAISE)*, **1**(1):5–14, 2009. 117

[85] STEPHEN KEEGAN, GREGORY MP O HARE, AND MICHAEL J O GRADY. **Easishop: Ambient intelligence assists everyday shopping**. *Information Sciences Journal*, **178**(3):588–611, 2008. 117

[86] DANTE I TAPIA, JUAN A FRAILE, SARA RODRÍGUEZ, RICARDO S ALONSO, AND JUAN M CORCHADO. **Integrating hardware agents into an enhanced multi-agent architecture for Ambient Intelligence systems**. *Information Sciences Journal*, **222**:47–65, 2013. 117

[87] EMILIO SERRANO AND JUAN BOTIA. **Validating ambient intelligence based ubiquitous computing systems by means of artificial societies**. *Information Sciences Journal*, **222**:3–24, 2013. 117

[88] TILL C LECH AND LEENDERT WM WIENHOFEN. **AmbieAgents: a scalable infrastructure for mobile and context-aware information services**. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 625–631. ACM, 2005. 117

[89] ICHIRO SATOH. **Mobile agents for ambient intelligence**. In *Massively Multi-Agent Systems I*, pages 187–201. Springer, 2005. 117

[90] DANTE I TAPIA AND JUAN MANUEL CORCHADO RODRÍGUEZ. **An ambient intelligence based multi-agent system for Alzheimer health care**. 2009. 117, 131

[91] JUAN M CORCHADO, JAVIER BAJO, AND AJITH ABRAHAM. **GerAmi: Improving healthcare delivery in geriatric residences**. *Intelligent Systems, IEEE*, **23**(2):19–25, 2008. 117, 131

[92] M. DE WEERDT AND B. CLEMENT. **Introduction to Planning in Multiagent Systems**. *Multiagent and Grid Systems*, **5**(4):345–355, 2009. 118

[93] ANTONIS BIKAKIS, GRIGORIS ANTONIOU, AND PANAYIOTIS HASAPIS. **Strategies for contextual reasoning with conflicts in ambient intelligence**. *Knowledge and Information Systems*, **27**(1):45–84, 2011. 118

[94] B. VERHEIJ. **Artificial argument assistants for defeasible argumentation**. *Artificial Intelligence Journal*, **150**(1):291–324, 2003. 118

[95] PAVLOS MORAITIS AND NIKOLAOS SPANOUDAKIS. **Argumentation-based agent interaction in an ambient-intelligence context**. *Intelligent Systems, IEEE*, **22**(6):84–93, 2007. 119

[96] ALI BAHRANI, JUN YUAN, CHUKWUEMEKA DAVID EMELE, DANIELE MASATO, TIMOTHY J NORMAN, AND DAVID MOTT. **Collaborative and context-aware planning**. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008. 119

[97] A BAHRANI AND JUN YUAN. **Extending plan representation with context-aware and seamless interoperability**. In *Conference of the International Technology Alliance*, pages 1–5, 2011. 119

[98] GEORGE FERGUSON, JAMES F ALLEN, BRADFORD W MILLER, ET AL. **TRAINS-95: Towards a Mixed-Initiative Planning Assistant.** In *Artificial Intelligence Planning Systems (AIPS)*, pages 70–77, 1996. 119

[99] J. FERBER. *Multi-agent systems: an introduction to distributed artificial intelligence*. 1999. 120

[100] W. PEDRYCZ AND P. RAI. **A multifaceted perspective at data analysis: a study in collaborative intelligent agents**. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, **38**(4):1062–1072, 2008. 120

[101] E.H. DURFEE AND V.R. LESSER. **Partial global planning: A coordination framework for distributed hypothesis formation**. *Systems, Man and Cybernetics, IEEE Transactions on*, **21**(5):1167–1183, 1991. 120

[102] K. DECKER AND V.R. LESSER. **Generalizing the Partial Global Planning Algorithm**. *International Journal of Cooperative Information Systems*, **2**(2):319–346, 1992. 120

[103] ALICE TONIOLO, TIMOTHY J NORMAN, AND KATIA P SYCARA. **An Empirical Study of Argumentation Schemes for Deliberative Dialogue.** In *European Conference on Artificial Intelligence (ECAI)*, pages 756–761, 2012. 120, 141

[104] A. TORREÑO, E. ONAINDÍA, AND O. SAPENA. **A Flexible Coupling Approach to Multi-Agent Planning under Incomplete Information**. *Knowledge and Information Systems*, **38**(1):141–178, 2014. 121

[105] ALESSANDRO RICCI, MIRKO VIROLI, AND ANDREA OMICINI. **Programming MAS with artifacts**. In *Programming multi-agent systems*, pages 206–221. Springer, 2006. 124

[106] MARCELA D RODRÍGUEZ, JESUS FAVELA, ALFREDO PRECIADO, AND AURORA VIZCAÍNO. **Agent-based ambient intelligence for healthcare**. *AI Commun.*, **18**(3):201–216, 2005. 131

[107] SUNGMEE PARK AND SUNDARESAN JAYARAMAN. **Enhancing the quality of life through wearable technology**. *Engineering in Medicine and Biology Magazine, IEEE*, **22**(3):41–48, 2003. 131

[108] STEVEN MINTON, JOHN BRESINA, AND MARK DRUMMOND. **Total-order and partial-order planning: A comparative analysis**. *Journal of Artificial Intelligence Research*, 1994. 139

[109] I. GARCÍA, S. PAJARES, L. SEBASTIÁ, AND E. ONAINDÍA. **Preference elicitation techniques for group recommender systems**. *Information Sciences Journal*, **189**:155–175, 2012. 153

[110] ALFONSO GEREVINI AND DEREK LONG. **Preferences and soft constraints in PDDL3**. In *International Conference on Automated Planning and Scheduling (ICAPS) workshop on planning with preferences and soft constraints*, pages 46–53, 2006. 153

[111] J.L. POLLOCK. **Defeasible reasoning with variable degrees of justification**. *Artificial Intelligence Journal*, **133**(1):233–282, 2001. 159

[112] D.R. GARCÍA. *Planificación y Formalización de Acciones para Agentes Inteligentes*. PhD thesis, University of Nacional del Sur, Bahía Blanca, Argentina (in Spanish), 2011. 164, 168, 170, 175

[113] M. THIELSCHER. **The Qualification Problem: A solution to the problem of anomalous models**. *Artificial Intelligence Journal*, **131**(1-2):1–37, 2001. 165

[114] DAVID E. SMITH AND MARK A. PEOT. **Postponing Threats in Partial-Order Planning**. In *National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993.*, pages 500–506, 1993. 172

[115] ALEJANDRO TORREÑO, EVA ONAINDIA, AND OSCAR SAPENA. **FMAP: Distributed cooperative multi-agent planning**. *Applied Intelligence Journal*, **41**(2):606–626, 2014. 179, 184, 185

[116] DANIEL BRYCE AND SUBBARAO KAMBHAMPATI. **A Tutorial on Planning Graph Based Reachability Heuristics**. *AI Magazine*, **28**(1):47–83, 2007. 183

[117] JUAN M ALBEROLA, JOSE M SUCH, VICENT BOTTI, AGUSTÍN ESPINOSA, AND ANA GARCÍA-FORNES. **A scalable multiagent platform for large systems**. *Computer Science and Information Systems*, **10**(1):51–77, 2013. 187

[118] STELLA HERAS, VICENTE JULIÁN, AND VICENTE BOTTI. **CBR Contributions to Argumentation in MAS**. In *Innovations in Hybrid Intelligent Systems*, pages 304–311. Springer, 2007. 199