Context-Aware Multi-Agent Planning in Intelligent Environments

Sergio Pajares Ferrando and Eva Onaindia

Dpto. de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera, s/n. 46022 Valencia, Spain
E-mail: {spajares,onaindia}@dsic.upv.es

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.

Keywords:
Context-aware reasoning, ambient intelligence, multi-agent planning, defeasible argumentation.

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[6, 23]. 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 [13] 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 [40].

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, technology 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 [1, 2]. As an example, Easishop is an ubiquitous commerce application for assistance in everyday shopping that emerges as a juxtaposition of AmI and e-commerce [24]. 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 [8]. 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 [44] or in multi-agent based simulations for testing and validating large-scale AmI systems in dangerous environments [41]. 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 [26] for context-based information services, or the SpacialAgents platform [39] 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 [43]. Homecare applications are becoming very popular, above all among elderly people [10], 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 [21] 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 [11] 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 [3] where device agents do not take part in the planning activity but they are simply device agents. Therefore, agents in [3] 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 [9]. We opt for using *argumentation* [49] 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. 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 [8] and P. Moraitis [28]. 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 [28], 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 [18] combines argumentation (as a mechanism for reasoning about context) and a centralized planner that follows a Partial-Order Planning (POP) approach [34] although this framework has not been experimentally tested in real applications yet. Unlike [18], F. Amigoni presents a planner applied to an AmI environment in the field of health-care [3], 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 [3], however, does not use argumentation. On the other hand, the work of A. Bahrani [4, 5], which follows a mixed initiative planning approach [16], 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 [15, 33]. 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 [7] 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 [14] and its extension, the generalized partial global planning approach [12]. In addition, A. Toniolo [45, 46] 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 3, the works in [30, 31, 32] 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 [47, 48]. 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 [47] 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 1 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 1: Summary of the related work.

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 [35]. 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.

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 [17] 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 [34], 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 [42], 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 [18], 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 [18].

The works in [30, 31, 32] 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 [30, 31, 32] 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.

4. Definition of Components of the Context-Aware System

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

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 [36]. 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.

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 $\langle v_i, vl_i \rangle$.

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:

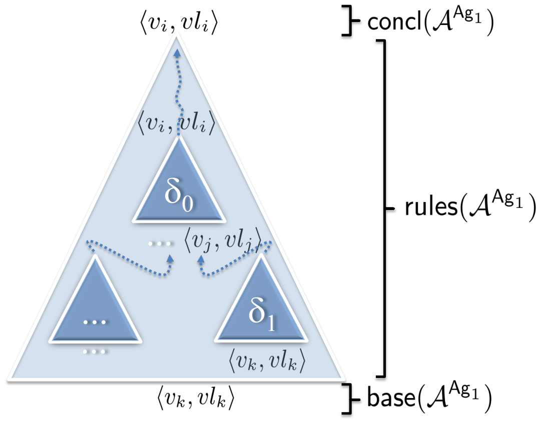
1. $\mathsf{AG} = \{\mathsf{Ag}_1 \ldots \mathsf{Ag}_n\} \mid n = |AG|$ is a finite non-empty set of ambient agents with planning and argumentation capabilities. It is assumed that agents are fully cooperative.

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.

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 [18, 31, 32]). In the rest of the paper, we will use letters $\mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z}$ to denote arguments, and alpha symbols $\alpha_1, \alpha_2, \ldots \alpha_n$, 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 1, 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 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.



$\langle v_i, vl_i \rangle$
concl($\mathcal{A}^{\mathsf{Ag}_1}$)
$\langle v_i, vl_i \rangle$
δ_0
rules($\mathcal{A}^{\mathsf{Ag}_1}$)
... $\langle v_j, vl_j \rangle$
...
δ_1
...
$\langle v_k, vl_k \rangle$
$\langle v_k, vl_k \rangle$
base($\mathcal{A}^{\mathsf{Ag}_1}$)

Figure 1: 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.

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 $\Pi = (\mathsf{A}(\Pi), \mathsf{AR}(\Pi), \mathsf{G}(\Pi), \mathsf{OC}(\Pi), \mathsf{CL}(\Pi), \mathsf{SL}(\Pi))$, where $\mathsf{A}(\Pi)$ denotes the set of action steps, $\mathsf{AR}(\Pi)$ represents the set of argument steps, $\mathsf{G}(\Pi)$ is the set of goals of the planning task (the user's needs), $\mathsf{OC}(\Pi)$ is a set of ordering constraints between actions in $\mathsf{A}(\Pi)$ (denoted by the relation $\prec$), and $\mathsf{CL}(\Pi)$ and $\mathsf{SL}(\Pi)$ 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 $\mathcal{A}^{\mathsf{Ag}_1} \in \mathsf{AR}(\Pi)$, i.e., $\langle v_k, vl_k\rangle \in \mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ (Figure 2). 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, which introduces the **support link** $(\mathcal{A}^{\mathsf{Ag}_1}, \langle v_i, vl_i\rangle, \alpha_{\mathsf{G}}) \in \mathsf{SL}(\Pi)$, where $\mathsf{SL}(\Pi) \subseteq \Delta \times \mathsf{G}(\Pi) \times \mathsf{A}$. In contrast, the goal $\langle v_k, vl_k\rangle$ must be supported by an action, α_1 in Figure 2, which introduces the **causal link** $(\alpha_1, \langle v_k, vl_k\rangle, \mathcal{A}^{\mathsf{Ag}_1}) \in \mathsf{CL}(\Pi)$, where $\mathsf{CL}(\Pi) \subseteq \mathsf{A} \times \mathsf{G}(\Pi) \times \Delta$. Specifically, in Figure 2, 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 $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \langle v_i, vl_i\rangle$; $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1}) = \{\langle v_i, vl_i\rangle \prec \langle v_k, vl_k\rangle\}$; 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* [22], 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 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 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, $\mathcal{A}^{\mathsf{Ag}_1}$, 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.

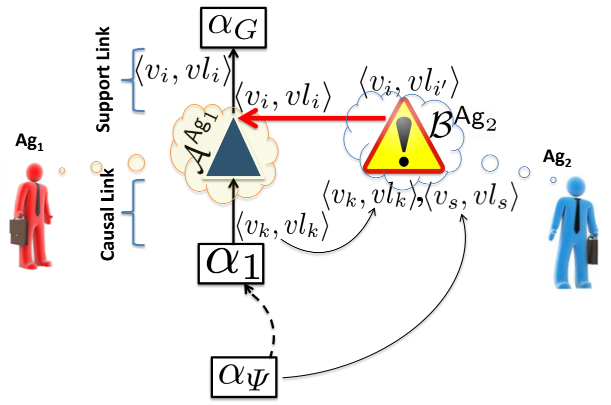

α_G
Support Link
$\langle v_i, vl_i \rangle$
$\langle v_i, vl_i \rangle$
$\langle v_i, vl_{i'} \rangle$
$\mathcal{B}^{\mathsf{Ag}_2}$
$\mathcal{A}^{\mathsf{Ag}_1}$
Ag1
Ag2
Causal Link
$\langle v_k, vl_k \rangle, \langle v_s, vl_s \rangle$
$\langle v_k, vl_k \rangle$
α_1
α_Ψ

Figure 2: 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 [31].

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 [10, 37, 43]. 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 [38]. 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 [38], 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.

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 [25], 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.

5.1.1. 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 1, 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 1: 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).

5.1.2. State Variables

In PDDL3.1, state variables are specified as functions with any number of parameters. Listing 2 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, 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: Single-valued variables represented as functions.

Additionally, particular pieces of information are modelled with predicates as shown in Listing 3. 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 3: 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 4 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 4: Multi-valued variables represented as multi-functions.

5.1.3. Defeasible Rules and Actions

Listing 5 and listing 6 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 5 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 4.5). Listing 6 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 5: 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 6: The body of the defeasible rule matches the effects of the action moving-BLS-medical-assistance to deal with the qualification problem.

5.1.4. Objects, Initial State and Goals

Listing 1 displays the object types of our health-care scenario. Particularly, in our problem we model three hospitals in a city {`H1, H2, H3`} (Listing 7). 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 7: 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 8 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 8 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 8: 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 9.

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

Listing 9: Global goals.

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 [27]. 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 3). Then it comes the planning stage (Plan Proposal Process in Figure 3) 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 3) 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 3) and they continue the search exploration. The process is repeated until a solution plan is found.

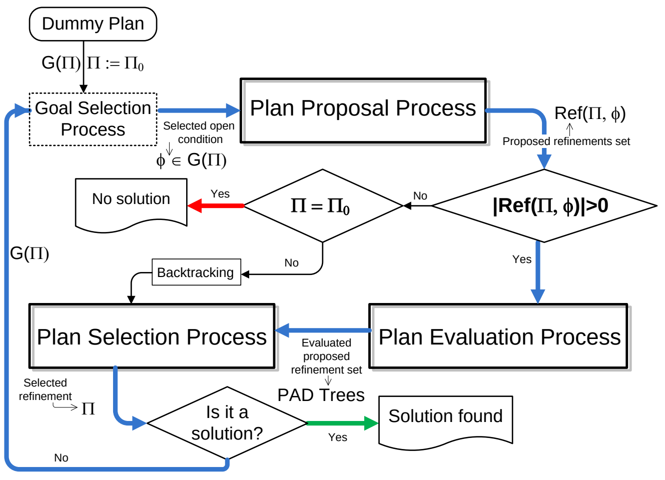


Figure 3: Overview of the CAMAP protocol.

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 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 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 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 5 and 6, respectively, among others. As shown Figure 4(a), one of the refinements proposed by agent Ag_1 is $\Pi_r \in \mathsf{Ref}(\Pi, \mathsf{P}(\alpha_{\mathsf{G}}))$ such that $\mathsf{OC}(\Pi_r) = \{\alpha_\Psi \prec \alpha_1;\ \alpha_1 \prec \mathcal{A}^{\mathsf{Ag}_1};\ \mathcal{A}^{\mathsf{Ag}_1} \prec \alpha_{\mathsf{G}}\}$, 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* $\rangle\}$.

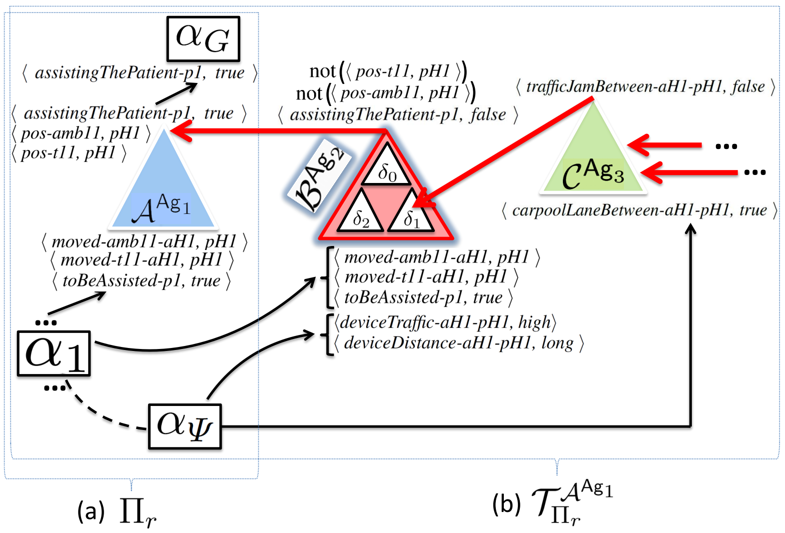

α_G
⟨ assistingThePatient-p1, true ⟩
⟨ assistingThePatient-p1, true ⟩
⟨ pos-amb11, pH1 ⟩
⟨ pos-t11, pH1 ⟩
$\mathcal{A}^{\mathsf{Ag}_1}$
⟨ moved-amb11-aH1, pH1 ⟩
⟨ moved-t11-aH1, pH1 ⟩
⟨ toBeAssisted-p1, true ⟩
α_1
α_Ψ
not(⟨ pos-t11, pH1 ⟩)
not(⟨ pos-amb11, pH1 ⟩)
⟨ assistingThePatient-p1, false ⟩
$\mathcal{B}^{\mathsf{Ag}_2}$
δ_0
δ_2
δ_1
⟨ trafficJamBetween-aH1-pH1, false ⟩
$\mathcal{C}^{\mathsf{Ag}_3}$
⟨ carpoolLaneBetween-aH1-pH1, true ⟩
⟨ moved-amb11-aH1, pH1 ⟩
⟨ moved-t11-aH1, pH1 ⟩
⟨ toBeAssisted-p1, true ⟩
⟨deviceTraffic-aH1-pH1, high⟩
⟨ deviceDistance-aH1-pH1, long ⟩
(a) Π_r
(b) $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$

Figure 4: 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 4(a)).

In most MAP systems [7, 45, 46], open goals are supported by the inclusion of an action whose effects match the open goals. However, as we said in Section 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.

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 $\mathcal{A}^{\mathsf{Ag}_i} \in \mathsf{AR}(\Pi_r)$ 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 $\mathcal{A}^{\mathsf{Ag}_i} \in \mathsf{AR}(\Pi_r)$.
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 [17]. 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 [17]: 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 6.1. This is graphically shown in Figure 4(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* ⟩*)*}, $\{\delta_0; \delta_1; \delta_2\}$) that attacks $\mathcal{A}^{\mathsf{Ag}_1}$, as shown in Figure 4(b); this attack is derived from the defeasible rules in Listing 10 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 10: Defeasible rules of agent Ag_2 for representing situations in which the ambulance may not arrive on time.

Assuming that {⟨ *deviceTraffic-aH1-pH1, high* ⟩, ⟨ *deviceDistance-aH1-pH1, long* ⟩} $\subseteq \Psi_{\mathsf{Ag}_2}$, as shown in Listing 8 of Section 5.1.4, 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 11). 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 4(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 11: 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 5.1.4), 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.

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 [29]. 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.

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.

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 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.

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.

7.2.1. Performance and Scalability

For evaluating the performance and scalability, we measured the computational time of CAMAP and TMAP to find a solution plan. Figure 5 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:

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

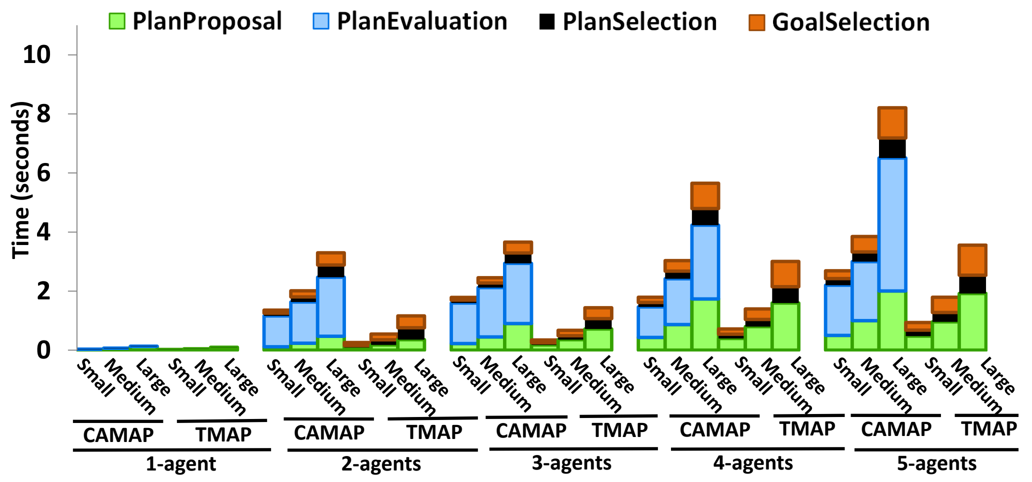


Figure 5: Evaluating the average time spent on each stage.

(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 5 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 6 corroborates this fact. The horizontal axis of Figure 6 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 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.

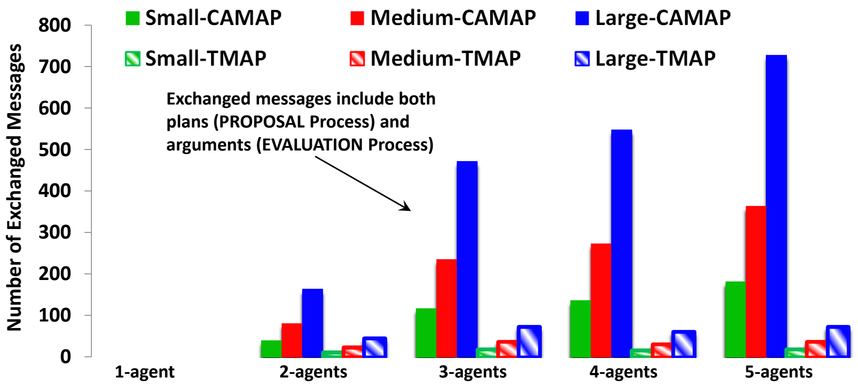


Figure 6: Evaluating the total number of exchanged messages between ambient agents.

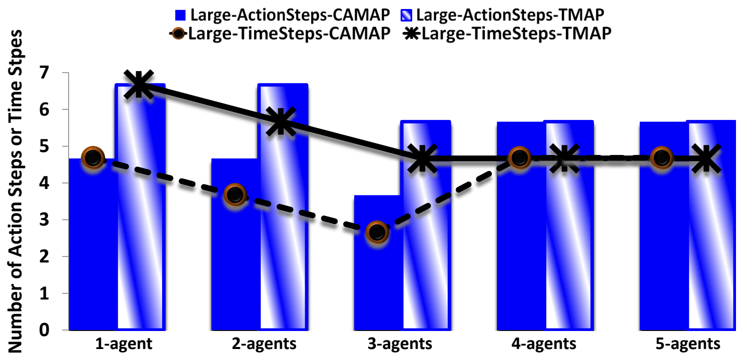


Figure 7: Evaluating the quality of the solution plans obtained for large difficulty: average number of action steps and average number of time steps.

7.2.2. 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 7 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 7 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 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.

7.2.3. 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 8. As can be seen, TMAP solutions included at least 30% of action steps that CAMAP agents acknowledged not to be success-

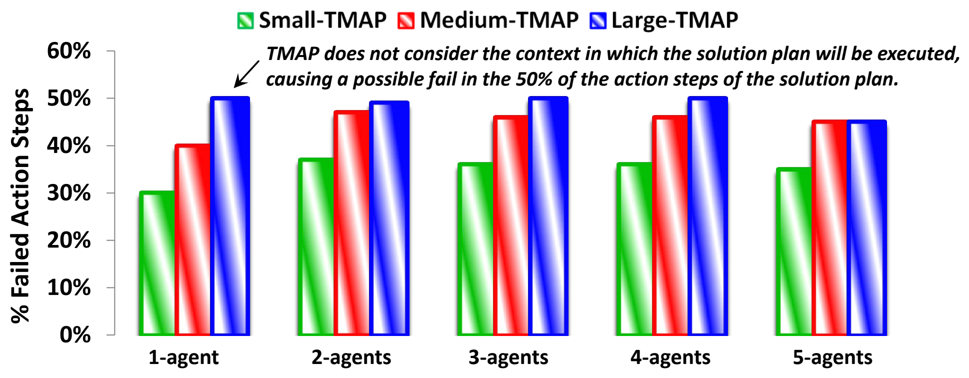


Figure 8: Evaluating the % of failing actions of the obtained solution plans.

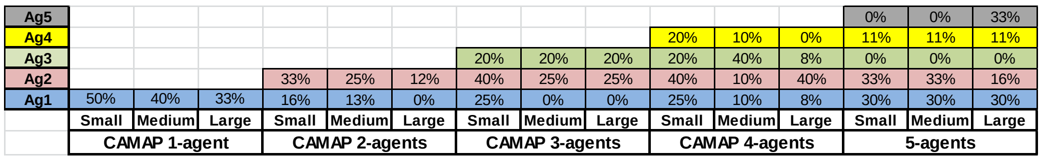
fully executed. For medium and large size problems, this percentage increases considerably. However, there are hardly differences in the number of failing actions between the agent teams, an indication that feasibility is not dependent on whom proposes the actions or how many agents a team has.

7.2.4. Trust and Contribution

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

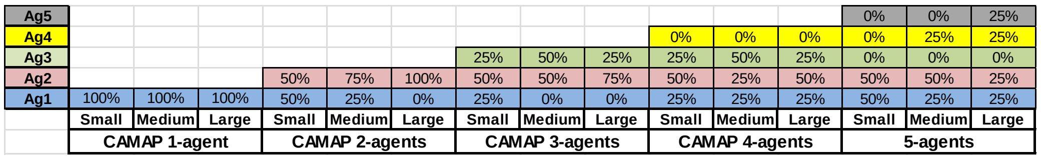
- The trust level for an agent is calculated as the number of undefeated argument steps in all of the plans proposed by an agent divided by the total number of argument steps proposed by the same agent in the Plan Proposal Process.
- The contribution level of an agent to a solution plan is calculated 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 9 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 9 indicate that the more defeasible rules known by the agents, the lower trust level achieved by the agents. For instance, in Figure 9 (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



	CAMAP 1-agent			CAMAP 2-agents			CAMAP 3-agents			CAMAP 4-agents			5-agents		
	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
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%

Figure 9: Evaluating the agents' trust level based on their proposed argument steps.



	CAMAP 1-agent			CAMAP 2-agents			CAMAP 3-agents			CAMAP 4-agents			5-agents		
	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
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%

Figure 10: Evaluating the agents' contribution level in the solution plans.

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 10 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 high individual trust level in this team. As we can see in Figure 9, Ag_2 has the highest individual trust level for this experiment.

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.

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 [19, 20]. 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.

References

[1] E. Aarts and R. Wichert. Ambient intelligence. *Technology Guide*, Springer, pp. 244–249, 2009.

[2] E. Aarts and B. de Ruyter. New research perspectives on Ambient Intelligence. *Journal of Ambient Intelligence and Smart Environments*, Springer, pp. 1(1):5–14, 2009.

[3] 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*, pp. 35(1):7–21, 2005.

[4] A. Bahrani, J. Yuan, C.D. Emele, D. Masato, T.J. Norman and D. Mott. Collaborative and context-aware planning. In Proc. of the *Military Communications Conference*, pp. 1–7, 2008.

[5] A. Bahrani and J. Yuan. Extending Plan Representation with Context-Aware and Seamless Interoperability. In Proc. of the *Conference of the International Technology Alliance*, pp. 1–5, 2011.

[6] M. Baldauf, S. Dustdar and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, pp. 2(4):263–277, 2007.

[7] A. Belesiotis and M. Rovatsos and I. Rahwan. Agreeing on Plans Through Iterated Disputes. In Proc. of the *9th Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2010)*, pp. 765–772, 2010.

[8] A. Bikakis and G. Antoniou. Defeasible Contextual Reasoning with Arguments in Ambient Intelligence. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1492–1506, 2010.

[9] A. Bikakis, G. Antoniou and P. Hassapis. Strategies for contextual reasoning with conflicts in ambient intelligence. *Knowledge and Information Systems*, pp. 27(1):45–84, 2011.

[10] J.M. Corchado, J. Bajo and A. Abraham. GerAmi: Improving healthcare delivery in geriatric residences. *Intelligent Systems, IEEE*, pp. 23(2):19–25, 2008.

[11] M. de Weerdt and B. Clement. Introduction to Planning in Multiagent Systems. *Multiagent and Grid Systems*, pp. 5(4):345–355, 2009.

[12] K. Decker and V.R. Lesser. Generalizing the Partial Global Planning Algorithm. *International Journal of Cooperative Information Systems*, pp. 2(2):319–346, 1992.

[13] A.K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, pp. 5(1):4–7, 2001.

[14] E.H. Durfee and V.R. Lesser. Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, pp. 21(5):1167–1183, 1991.

[15] J. Ferber. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. *Addison-Wesley New York*, 222, 1999.

[16] G. Ferguson, J. Allen and B. Miller. TRAINS-95: Towards a mixed-initiative planning assistant. In Proc. of the *3rd Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pp. 70–77, 1996.

[17] A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, pp. 4(2):95–138, 2004.

[18] D. García, A. García, and G. Simari. Defeasible Reasoning and Partial Order Planning. In Proc. of the *International Conference on Foundations of information and knowledge systems, FoIKS 2008*, LNCS 4932, pp. 311–328, 2008.

[19] I. García, S. Pajares, L. Sebastiá and E. Onaindia. Preference elicitation techniques for group recommender systems. *Information Sciences*, pp. 189: 155–175, 2012.

[20] A. Gerevini and D. Long. Preferences and soft constraints in PDDL3. In Proc. of the *ICAPS Workshop on Planning with Preferences and Soft Constraints*, pp. 46–53, 2006.

[21] M. Ghallab, D.S. Nau and P. Traverso. Automated Planning: theory and practice. *Morgan Kaufmann*, 2004.

[22] M.L. Ginsberg and D.E. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, pp. 3:311–342, 1998.

[23] J. Hong, E. Suh, and S.J. Kim. Context-Aware systems: A literature review and classification. *Expert Systems with Applications*, pp. 36(4):8509–8522, 2009.

[24] S. Keegan, G.M.P. O'Hare, and M.J. O'Grady. Easishop: Ambient intelligence assists everyday shopping. *Information Sciences*, pp. 178(3):588–611, 2008.

[25] D.L. Kovacs. Complete BNF description of PDDL3.1. *Technical report*, 2011.

[26] T.C. Lech and L.W.M. Wienhofen AmbieAgents: a scalable infrastructure for mobile and context-aware information services. In Proc. of the *4th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pp. 625-631, 2005.

[27] S. Minton, J. Bresina and M. Drummond. Total-order and partial-order planning: A comparative analysis. *Journal of Artificial Intelligence Research*, pp. 2:227–262, 1994.

[28] P. Moraitis and N. Spanoudakis. Argumentation-based agent interaction in an ambient-intelligence context. *Intelligent Systems, IEEE*, pp. 22(6):84–93, 2007.

[29] X.L. Nguyen, and S. Kambhampati. Reviving partial order planning. In Proc. of the *International Joint Conferences on Artificial Intelligence (IJCAI 2001)*, pp. 17:459–466, 2001.

[30] S. Pajares and E. Onaindía. Defeasible Planning through Multi-Agent Argumentation. *Modelling Machine Emotions For Realizing Intelligence, Smart Innovation Systems and Technologies Series*, pp. 13:311–342, 2011.

[31] S. Pajares, E. Onaindía and A. Torreño. An Architecture for Defeasible-Reasoning-Based Cooperative Distributed Planning. In Proc. of the *19th International Conference on Cooperative Information Systems (CoopIS 2011), in conjunction with On the Move to Meaningful Internet Systems (OTM 2011)*, pp. 200–217, 2011.

[32] P. Pardo and S. Pajares and E. Onaindía and L. Godo and P. Dellunde. Multiagent Argumentation for Cooperative Planning in DeLP-POP. In Proc. of the *10th Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2011)*, pp. 971–978, 2011.

[33] 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*, pp. 38(4): 1062–1072, 2008.

[34] J.S. Penberthy and D.S. Weld. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In Proc. of the *International Conference on Principles of Knowledge Representation and Reasoning*, pp. 103–114, 1992.

[35] H. Prakken and G. Vreeswijk. Logics for Defeasible Argumentation. *Handbook of philosophical logic*, pp. 4:218–319, 2002.

[36] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. *Programming Multi-Agent Systems*, pp. 206–221, 2006.

[37] M.D. Rodríguez, J. Favela, A. Preciado, A. Vizcaíno. Agent-based ambient intelligence for healthcare. *AI Communications*, pp. 18(3):201–216, 2005.

[38] M.D. Rodríguez, J. Favela, A. Preciado, A. Vizcaíno. Enhancing the quality of life through wearable technology. *Engineering in Medicine and Biology Magazine, IEEE*, pp. 22(3):41–48, 2003.

[39] I. Satoh Mobile Agents for Ambient Intelligence. In Proc. of the *MMAS*, pp. 187-201, 2004.

[40] A. Schmidt. Ubiquitous computing - computing in context. *Personal and Ubiquitous Computing*, pp. 1-294, 2003.

[41] E. Serrano and J. Botia. Validating ambient intelligence based ubiquitous computing systems by means of artificial societies. *Information Sciences*, 2010.

[42] D.E. Smith, J. Frank and A.K. Jónsson. Bridging the gap between planning and scheduling. *The Knowledge Engineering Review*, pp. 15(1):47–83, 2000.

[43] D.I. Tapia and J.M. Corchado. An Ambient Intelligence Based Multi-Agent System for Alzheimer Health Care. *International Journal of Ambient Computing and Intelligence (IJACI)*, pp. 1(1):15–26, 2009.

[44] D. Tapia, J.A. Fraile, S. Rodriguez, R.S. Alonso, and J.M. Corchado. Integrating hardware agents into an enhanced multi-agent architecture for Ambient Intelligence systems. *Information Sciences*, 2011.

[45] A. Toniolo, T.J. Norman, and K. Sycara. Argumentation schemes for collaborative planning. In Proc. of the *14th Conference on Principles and Practice of Multi-Agent Systems*, 2011.

[46] A. Toniolo, T.J. Norman, and K. Sycara. An Empirical Study of Argumentation Schemes for Deliberative Dialogue. In Proc. of the *20th European Conference on Artificial Intelligence (ECAI)*, 242:756–761, 2012.

[47] A. Torreño, E. Onaindia and O. Sapena. An approach to MultiAgent Planning with Incomplete Information. In Proc. of the *20th European Conference on Artificial Intelligence (ECAI)*, 242:762–767, 2012.

[48] A. Torreño, E. Onaindia and O. Sapena. A Flexible Coupling Approach to Multi-Agent Planning under Incomplete Information. *Knowledge and Information Systems*, In Press, 2012.

[49] B. Verheij. Artificial argument assistants for defeasible argumentation. *Artificial Intelligence*, pp. 150(1-2):291–324, 2003.