UNIVERSIDAD POLITÉCNICA DE VALENCIA

Dpto. Sistemas Informáticos y Computación

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

TESIS DE MÁSTER

Multiagent Argumentation on Cooperative Planning in DeLP-POP

Autor: Sergio Pajares Ferrando

Dirigida por: Dra. Eva Onaindia de la Rivaherrera

Grupo de Tecnología Informática - Inteligencia Artificial (GTI-IA)
Group of Reasoning on Planning and Scheduling (GRPS-AI)
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia, Spain

30 de Noviembre de 2010

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

A mis padres, Claudio y Pilar.

Acknowledgements

I would like to thank all people who have helped and inspired me during my master thesis project.

I especially want to thank my supervisor, *Dr. Eva Onaindia*, whose encouragement, guidance and support from the initial to the final level enabled me to develop the present work. Her perpetual energy and enthusiasm in artificial intelligence research has motivated me. In addition, she was always accessible and willing to help me with the present master thesis project.

I would like to thank *Professor Vicente J. Botti* for let me be part of a Magentix2 development group, in which I learned a lot over the last two years.

Dr. Inmaculada García and *Dr. Laura Sebastiá* deserve a special thanks for introducing me to begin my research in this group two years ago.

My deepest gratitude goes to my family for their unflagging love and support throughout my life; this master thesis project is simply impossible without them. I am indebted to my parents, *Claudio Pajares* and *Mª Pilar Ferrando*, for their care and love.

Last but not least, thanks to *Irina Dionisio*, who always supported me and given me encouragement.

Contents

1 Introduction **9**
- 1.1 Motivation . . . 9
- 1.2 Objectives . . . 11
- 1.3 Work overview . . . 12

2 Preliminaries **13**
- 2.1 Artificial Intelligence . . . 13
  - 2.1.1 Intelligent agents . . . 13
  - 2.1.2 Multi-agent systems . . . 14
- 2.2 Planning . . . 15
  - 2.2.1 Partial Order Planning (POP) . . . 16
- 2.3 Argumentation . . . 18
  - 2.3.1 Defeasible argumentation in DeLP . . . 18
- 2.4 Argumentation in Planning . . . 26
  - 2.4.1 A DeLP extension for POP planning . . . 26

3 Multi-agent DeLP-POP **31**
- 3.1 Concepts for a multi-agent DeLP-POP . . . 31
  - 3.1.1 The planning domain of the agents . . . 32
  - 3.1.2 Proto-states . . . 33
  - 3.1.3 Pre-arguments . . . 35
  - 3.1.4 Cost of the actions and inferences . . . 35
  - 3.1.5 Agent's learning ability . . . 36
  - 3.1.6 Absolute and non-absolute threats . . . 36
- 3.2 Overview of Cooperative Planning . . . 39

3.2.1 Extending the single-agent DelP-POP to multi-agent DeLP-POP . . . 39
3.3 Dialogues for multi-agent DeLP-POP . . . 40
3.3.1 Dialogue-based plan evaluation . . . 42
3.3.2 Dialogue-based search in the space of plans . . . 44

4 Validation **51**
4.1 Representation of actions with defeasible effects . . . 51
4.2 Introduction to the scenario . . . 52
4.3 Scenario specification . . . 52
4.4 Searching for a solution plan . . . 55

5 Related Work **61**

6 Conclusions and Future Work **63**
6.1 Conclusions . . . 63
6.2 Future work . . . 64
6.3 Published papers . . . 64
6.3.1 Papers directly related to this work . . . 65
6.3.2 Other papers . . . 65

List of Figures

2.1 An argument $\mathcal{A}$ for ℓ made up of two rules δ_0, $\delta 1$. 21
2.2 An instantiated example. 22
2.3 Computing warrant for l: an undefeated argument exists. . . . 26
2.4 Threat types: (a) action-action, (b) action-argument and (c) argument-argument. 28
2.5 Solutions to (a), (b). Promote: (a'), (b'); and Demote: (a"), (b"). 29
2.6 Solutions to (c): Delay (c'), Defeat (c") and Disable (c"'). . . . 29

3.1 An argument $\mathcal{A}$ applicable at the proto-state S_α^Π. 34
3.2 How calculate the proto-states in the construction of the different plans. 34
3.3 An argument $\mathcal{B}$ refining an pre-argument $\mathcal{A}$ (I). 36
3.4 An argument $\mathcal{B}$ refining an pre-argument $\mathcal{A}$ (II). 37
3.5 Agents improving their beliefs and defeasible rules by means of dialogues. 38
3.6 A selected plan Π. 44
3.7 A derived plan $\Pi^{(n,Ann)}$. 45
3.8 Evaluating a selected plan Π. 46
3.9 A selected plan Π. 48
3.10 A refinement plan $\Pi_r^{(n,i)}$. 49
3.11 Agents proposing different plans. 50

4.1 Scenario of the application example 53

4.2 Different partial plans for the example scenario (I). (a), (b): Joe's turns and (c): Ann's turn. 57
4.3 Different partial plans for the example scenario (II). (d): Joe's turn. 58
4.4 Search in the space of partial-order plans for the example scenario. 60

1

Introduction

1.1 Motivation

Planning is the art of building control algorithms that synthesize a course of action to achieve a desired set of goals. The mainstream in planning is that of using heuristic functions to evaluate goals and choices of action or states on the basis of their expected utility to the planning agent [Ghallab et al., 2004]. In classical planning, intelligent agents must be able to set goals and achieve them, they have a perfect and complete knowledge of the world, and they assume their view of the world can only be changed through the execution of the planning actions. However, in many real-world applications, agents often have contradictory information about the environment and their deductions are not always certain information, but *plausible*, since the conclusions can be withdrawn when new pieces of knowledge are posted by other agents.

Multi-Agent Planning (MAP) generalizes the problem of planning in domains where several agents plan and act together, and have to share resources, activities, and goals. In a cooperative approach, where the agents are assumed to be cooperative, the emphasis is placed on how planning can be extended to a distributed environment. The planning agents of a MAP task exchange information about their plans, which they iteratively refine and

revise until they fit together [desJardins et al., 1999]. Typically, research in MAP has been more concerned with the design of distributed planning architectures, mechanisms for plan coordination, or solutions for merging the resulting local plans of agents into a global plan [Durfee, 1999, Durfee, 2001, Cox et al., 2005, de Weerdt et al., 2005]. Unlike these approaches, which emphasize the problem of controlling and coordinating a posteriori local plans of independent agents, we propose an argumentation mechanism to allow agents to jointly devise a global shared plan and carry out collective actions. In our proposal we allow agents to plan concurrent actions through the adoption of a partial-order planning (POP) paradigm [Barrett and Weld, 1994b].

On the other hand, argumentation, which has recently become a very active research field in computer science [Bench-Capon and Dunne, 2007], can be viewed as a powerful tool for reasoning about inconsistent information through a rational interaction of arguments for and against some conclusion. Systems that build on defeasible argumentation apply theoretical reasoning for the generation and evaluation of arguments, and they are used to build applications that deal with incomplete and contradictory information in dynamic domains ([Prakken and Sartor, 1997, García and Simari, 2004, Prakken et al., 1997, Rahwan and Amgoud, 2006]).

Argumentation-based frameworks have been used for reasoning about what actions are the best to be executed by an agent in a given situation. Argumentation has been applied on reasoning about conflicting plans and for generating consistent sets of intentions from a contradictory set of desires [Amgoud, 2003, Hulstijn and van der Torre,]. The work in [García and Simari, 2004] presents a defeasible argumentation framework for the definition of actions and the combination of these actions into plans. Recently, a number of attempts have been made to use argumentation to capture practical reasoning, that is reasoning about which actions are the best for a particular agent to do in a given situation [Rahwan and Amgoud, 2006]. Other approaches like [Atkinson and Bench-Capon, 2007] propose undertaking practical reasoning through the instantiation of an argument scheme and associated critical questions [Walton, 1996, Atkinson et al., 2006]. None of these works, however, apply to a multi-agent scenario. On the other hand, some extensions to cooperative agents can be found in the work [Belesiotis et al., 2010], an argumentation-based approach for coordinating several agents who discuss plan proposals in the language of situation calculus.

Particularly, the application of an argumentation-based formalism to deal with the defeasible nature of reasoning during the construction of a plan has been addressed by Garcia and Simari [García et al., 2008].

We support that a model for argumentation-based **multi-agent** planning where different agents are able to exchange reasons for or against how

to support a open goal of the plan, could significantly improve the currently single-agent DeLP-POP framework. Despite all previous work, only one recent work [Thimm, 2009, Thimm,] has attempted to realize argumentation in multi-agent systems using defeasible reasoning but they are not particularly concerned with the task of planning.

1.2 Objectives

This contribution proposes a model for argumentation-based multi-agent planning, with a focus on cooperative scenarios. It is based on a **multi-agent extension of DeLP-POP**, partial order planning on top of argumentation-based defeasible logic programming.

This framework combines POP's minimal constraints on execution ordering (see [Penberthy and Weld, 1992a]), with DeLP inference based on interactions between arguments (see [García and Simari, 2004]). A DeLP-POP planner can enforce goals with a combination of actions and undefeated arguments, if their conditions (are known to) apply. The advantages of DeLP-POP towards reasoning about actions are clear: if planning techniques prevent the well-known frame problem, by getting rid of the need to explicitly represent what does not change after an action, DeLP-POP succeeds against the qualification problem as well, since rules can encode defeasible effects of actions. Arguments, though, are not like actions in that they apply even if unintended. Thus, arguments will not only occur to intentionally support some step of a plan, but also they will happen to defeat or defend some such supporting argument and the plan containing it. The main challenge presented by multi-agent DeLP-POP is collaborative plan search. We present some results about dialogues for argumentative plan search that apply to cooperative planning as a cooperative scenario. In this scenario, we have a group of agents aware of a common set of goals (hence trustable), but ignorant of others' abilities and beliefs, who must find a plan. Dialogues will be turn-based, since this choice models typically cooperative scenarios where all agents are treated in a uniform way. A dialogue consists in a series of exchanges of:

- Plan proposals addressing the current goal, and,
- Plan proposals involved in the discovering of new threats.

Atomic information (facts, rules, actions) contained in the above kind of exchanged messages will be extracted and learned by the rest of agents.

Summarizing, our main contribution is to extend Simari's work [García et al., 2008] and come up with the next features:

1. Collaborative plan search on DeLP-POP,
2. Argumentative dialogues for plan search, and,
3. Validation on cooperative scenarios.

1.3 Work overview

We present the main structure of this thesis at this point. Apart from this first introductory chapter, the remaining chapters are organized as follows:

- Chapter 2 explains the fundamentals of Defeasible Logic and Partial Order Planning,
- Chapter 3 is the key chapter of this dissertation and presents our multi-agent argumentation model. An outline of the implemented algorithm will be also shown in this chapter.
- Chapter 4 presents an example of application to show the behaviour of our multi-agent argumentation model,
- Chapter 5 summarizes previous related work, and,
- Chapter 6 presents the conclusions and future work.

2

Preliminaries

2.1 Artificial Intelligence

Artificial intelligence (AI) is the intelligence of machines and the branch of computer science that aims to create it. AI textbooks[1] define the field as *the study and design of intelligent agents* where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success. John McCarthy, who coined the term in 1956, defines it as *the science and engineering of making intelligent machines.*

Two currently important topics in artificial intelligence are *intelligent agents* and their integration towards to the design and development *multi-agent systems*(MAS). In what follows, we introduce both topics.

2.1.1 Intelligent agents

In artificial intelligence, an intelligent agent [Wooldridge and Jennings, 1995, Ferber, 1999, Wooldridge, 2009] is an autonomous entity which observes and acts upon an environment (i.e. it is an agent) and directs its activity towards

[1] http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/Reference#texts

achieving goals (i.e. it is rational). Intelligent agents may also learn or use knowledge to achieve their goals. They may be very simple or very complex: a reflex machine such as a thermostat is an intelligent agent, as is a human being, as is a community of human beings working together towards a goal.

Intelligent agents are often described schematically as an abstract functional system similar to a computer program. For this reason, intelligent agents are sometimes called abstract intelligent agents to distinguish them from their real world implementations as computer systems, biological systems, or organizations. Some definitions of intelligent agents emphasize their autonomy, and so prefer the term autonomous intelligent agents. Still others [Russell and Norvig, 2009] considered goal-directed behavior as the essence of intelligence and so prefer a term borrowed from economics, rational agent.

Intelligent agents in artificial intelligence are closely related to agents in economics, and versions of the intelligent agent paradigm are studied in cognitive science, ethics, the philosophy of practical reason, as well as in many interdisciplinary socio-cognitive modeling and computer social simulations.

Intelligent agents are also closely related to software agents (an autonomous software program that carries out tasks on behalf of users). In computer science, the term intelligent agent may be used to refer to a software agent that has some intelligence, regardless if it is not a rational agent by Russell and Norvig's definition. For example, autonomous programs used for operator assistance or data mining (sometimes referred to as bots) are also called intelligent agents.

Hence summarizing the above, the features of an intelligent agent should be autonomous, reactive, proactive, able to communicate, adaptive, goal-oriented, capable to cooperate, reason and learn, and flexible [Ganzha et al., 2005].

2.1.2 Multi-agent systems

Quoting from [Ralston et al., 1993] multi-agent systems are computational systems in which several artificial agents, which are programs, interact or work together over a communications network to perform some set of tasks jointly or to satisfy some set of goals. These systems may consist of homogeneous or heterogeneous agents. Examples of agents would be ones for detecting and diagnosing network problems occurring on a segment of a local area network; for scheduling the activities of a group of machines in a workcell on a factory floor; or for locating agents that are selling a specific product and deciding on what price to pay. Agents may be characterized by whether they are benevolent cooperative (agents are assumed to be fully cooperative in this master thesis project) or self-interested (this topic appears to be a promising extension of this master thesis project according to

some reviewers). Cooperative agents work toward achieving a set of shared goals, whereas self-interested agents have distinct goals but may still interact to further their own goals. For example, in a manufacturing setting, where agents are responsible for scheduling different aspects of the manufacturing process, agents in the same manufacturing company would behave in a cooperative way, while agents representing two separate companies, where one company was outsourcing part of its manufacturing process to the other company, would behave in a self-interested way. Agents often need to be semi-autonomous and highly adaptive due to their open operating environments, where the configuration and capabilities of other agents and network resources change dynamically. Agent autonomy relates to an agent's ability to make its own decisions about what activities to do, when to do them, and to whom information should be communicated. Scientific research and practice in this area, which is also called Distributed Artificial Intelligence, focuses on the development of computational principles and models for constructing, describing, and analyzing the patterns of interaction and coordination in both large and small agent societies.

2.2 Planning

Planning is the art of building control algorithms that synthesize a course of action to achieve a desired set of goals. We consider planning problems encoded in a formal, first-order language such as STRIPS [Fikes and Nilsson, 1971], particularly in a propositional version of STRIPS. We will denote the set of all propositions by $\mathcal{P}$ (ground facts or literals). A planning state s is defined as a finite set propositions $s \subseteq \mathcal{P}$. A (grounded) planning task is a triple $\mathcal{T} = \langle A, \mathcal{I}, G \rangle$, where A is the set of deterministic actions of the agent's model that describes the state changes, and $\mathcal{I} \subseteq \mathcal{P}$ (the initial state) and $G \subseteq \mathcal{P}$ (the goals) are sets of propositions. An action $\alpha \in A$ is a tuple $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$, where $\mathsf{P}(\alpha) \subseteq \mathcal{P}$ is the set of propositions that represents the action's preconditions, and $\mathsf{X}(\alpha) \subseteq \mathcal{P}$ and $\overline{\mathsf{X}(\alpha)} \subseteq \mathcal{P}$ are the sets of propositions that represent the positive and negative effects, respectively. We will represent an action α as follows:

$$\{q_1, \ldots, q_n, \sim r_1, \ldots, \sim r_m\} \xleftarrow{id} \{p_1, \ldots, p_k\} \tag{1}$$

where id is the action name, $\forall_{i=1}^{k} p_i \in \mathsf{P}(\alpha)$, $\forall_{i=1}^{n} q_i \in \mathsf{X}(\alpha)$, and $\forall_{i=1}^{m} r_i \in \overline{\mathsf{X}(\alpha)}$. An action α is executable in state s if $\mathsf{P}(\alpha) \subseteq s$. The state resulting from executing α is defined as $s' = (s \setminus \overline{\mathsf{X}(\alpha)}) \cup \mathsf{X}(\alpha)$. That is, we delete any proposition in s that belongs to $\overline{\mathsf{X}(\alpha)}$, and add the propositions in $\mathsf{X}(\alpha)$. A solution plan (Π) for a planning task $\mathcal{T}$ is a set of actions $\Pi = \{\alpha_1, \ldots, \alpha_n\} \subseteq$

A such that when applied to $\mathcal{I}$, it leads to a final state in which the goals G are satisfied. A planning task $\mathcal{T}$ is solvable if there exists at least one plan for it.

2.2.1 Partial Order Planning (POP)

Forward and backward state-space search are particular forms of totally ordered plan search. They explore only strictly linear sequences of actions directly connected to the start or goal. This means that they cannot take advantage of problem decomposition. Rather than work on each subproblem separately, they must always make decisions about how to sequence actions from all the subproblems. We would prefer an approach that works on several subgoals independently, solves them with several subplans, and then combines the subplans.

Such an approach also has the advantage of flexibility in the order in which it constructs the plan. That is, the planner can work on obvious or important decisions first, rather than being forced to work on steps in chronological order. For example, a planning agent that is in Valencia and wishes to be in Monte Carlo might first try to find a flight from San Francisco to Paris; given information about the departure and arrival times, it can then work on ways to get to and from the airports.

In what follows, we provide a brief introduction to the Partial-Order Planning (POP) paradigm ([Barrett and Weld, 1994a][Penberthy and Weld, 1992a]). A more detailed tutorial can be found in [Weld, 1994]. In POP, search is done through the space of incomplete partially-ordered plans as opposite to state-based planning.

The general strategy of delaying a choice during search is called a least commitment strategy. There is no formal definition of least commitment, and clearly some degree of commitment is necessary, lest the search would make no progress. Despite the informality, least commitment is a useful concept for analyzing when decisions should be made in any search problem.

Note also the dummy actions called Start and Finish, which mark the beginning and end of the plan. Calling them actions symplifies things, because now every step of a plan is an action. We start with an empty plan. Then we consider ways of refining the plan until we come up with a complete plan that solves the problem. The actions in this search are not actions in the world, but actions on plans: adding a step to the plan, imposing an ordering that puts one action before another, and so on.

A key concept in POP is that of *partial-order plan.*

Definition 1. *A **partial-order plan** is a tuple* $\Pi = \langle A_\Pi, OC, CL, G, \mathsf{Threats}\rangle$, *where:*

- $A_\Pi \subseteq A$ *is the set of ground actions*[2] *in* Π,
- OC *is a set of ordering constraints* ($\prec$) *over* A,
- CL *is a set of causal links over* A. *A causal link is of the form* (α_i, p, α_j), *and denotes that the precondition* p *of action* α_j *will be supported by an* X *effect of action* α_i,
- G *is the set of* open conditions *of* Π. *Let* $\alpha_i \in A$; *if* $\exists p \in \mathsf{P}(\alpha_i) \wedge$ / $\exists \alpha_j \in A / (\alpha_j, p, \alpha_i) \subseteq CL$, *then* p *is said to be an open condition, and,*
- $\mathsf{Threats}$ *is the set of* unsafe causal links *of* Π, *also called the* threats. *Let* $(\alpha_i, p, \alpha_j) \subseteq CL$; (α_i, p, α_j) *is unsafe if there exists an action* $\alpha_k \in A$ *such that* $\overline{p} \in \mathsf{X}(\alpha_k)$ *and* $OC \cup \{\alpha_i \prec \alpha_k \prec \alpha_j\}$ *is consistent.*

Given a planning task $\mathcal{T} = \langle A, \mathcal{I}, G\rangle$, a POP algorithm starts with an empty partial plan and keeps refining it until a solution plan is found. The initial empty plan $\Pi^{(0)} = \langle A_\Pi, OC, CL, G, \mathsf{Threats}\rangle$ contains only two dummy actions $A_\Pi = \{\alpha_0, \alpha_G\}$, the *start* action α_0, and the *finish* action α_G, where $\mathsf{P}(\alpha_G) = G$, $\mathsf{X}(\alpha_0) = \mathcal{I}$, $\{\alpha_0 \prec \alpha_G\} \subseteq OC$, $CL = \emptyset$, $G = G$ and $\mathsf{Threats} = \emptyset$. The empty plan has no causal links or threats, but, has open condition corresponding to the preconditions of α_f (the top-level goals G). A refinement step in a POP algorithm involves two things; first, selecting a flaw (an open condition or a threat) in a partial plan Π, and then selecting a resolver for the flaw. The different ways of solving a flaw are:

- Supporting an open condition with an *action step*. If p is an open condition, an action α needs to be selected that achieves p. α can be a new action from A, or any action that already exists in A_Π. Solving an open condition involves adding a causal link to Π to record that p is achieved by the chosen action step.
- Solving a threat with an *ordering constraint*. When the flaw chosen is an unsafe causal link (α_i, p, α_j) that is threatened by an action α_k, it can be repaired either by adding the ordering constraint $\alpha_k \prec \alpha_i$, or the ordering constraint $\alpha_j \prec \alpha_k$, into OC. This solving method involves reordering the action steps in Π.

[2]Partial-order planners are capable of handling partially instantiated action instances and hence, the definition of a partial order plan typically includes a set of equality constraints on free variables in A [Penberthy and Weld, 1992a]. We will, however, restrict our attention to ground action instances without any loss of generality for our purposes.

Definition 2. *A plan* $\Pi = \langle A_\Pi, OC, CL, G, \mathsf{Threats}\rangle$ *is* ***complete*** *if it has no open conditions* ($G = \emptyset$).

Definition 3. *A plan* $\Pi = \langle A_\Pi, OC, CL, G, \mathsf{Threats}\rangle$ *is* ***conflict-free*** *if it has no unsafe causal links* ($\mathsf{Threats} = \emptyset$).

Definition 4. *A plan* $\Pi = \langle A_\Pi, OC, CL, G, \mathsf{Threats}\rangle$ *is a* ***solution*** *if it is complete and conflict-free.*

Compared with total-order planning, partial-order planning has a clear advantage in being able to decompose problems into subproblems. This is very useful in an environment where each agent is able to resolve a different problem. It is It also has a disadvantage in that it does not represent states directly, so it is harder to estimate how far a partial-order plan is from achieving a goal. At present, there is less understanding of how to compute accurate heuristics for partial-order planning than for total-order planning.

2.3 Argumentation

Argumentation is the interdisciplinary study of how humans should, can, and do reach conclusions through logical reasoning, that is, claims based, soundly or not, on premises. It includes the arts and sciences of civil debate, dialogue, conversation, and persuasion. It studies rules of inference, logic, and procedural rules in both artificial and real world settings.

Argumentation includes debate and negotiation which are concerned with reaching mutually acceptable conclusions. It also encompasses eristic dialog, the branch of social debate in which victory over an opponent is the primary goal. This art and science is often the means by which people protect their beliefs or self-interests in rational dialogue, in common parlance, and during the process of arguing. For instance, argumentation is used in law, in trials, in preparing an argument to be presented to a court, and in testing the validity of certain kinds of evidence. Also, argumentation scholars study the post hoc rationalizations by which organizational actors try to justify decisions they have made irrationally.

2.3.1 Defeasible argumentation in DeLP

In [García and Simari, 2004], the authors propose an argumentation-based approach for defeasible logic in single-agent contexts. In these contexts, an agent makes use of defeasible argumentation with an internal deliberation purpose. The basic information is initially given and assumed to remain fixed throughout internal argumenation; it is the set of acceptable

conclusions which may vary as this argumentation is carried on and new arguments are added into consideration. In this section, we summarize the main concepts of the work on Defeasible Logic Programming (DeLP), a formalism that combines Logic Programming and Defeasible Argumentation [García and Simari, 2004]. The basic elements in DeLP are facts and rules. Let $\mathcal{L}$ denote a set of literals, where a literal h is a fact A or a negated fact $\sim A$, and, the symbol $\backsim$ represents the strong negation. The set of rules is divided into *strict rules*, i.e. rules encoding strict consequences, and *defeasible rules*, which derive uncertain or defeasible conclusions. A strict rule is an ordered pair $head \leftarrow body$, and a defeasible rule is an ordered pair $head \prec body$, where $head$ is a literal, and $body$ is a finite non-empty set of literals. For example, the strict rule $animal \leftarrow bird$ is denoting the piece of information "a bird is an animal". However, a defeasible rule is used to describe tentative knowledge that may be used if nothing else can be posed against it, e.g. "birds fly" (fly $\prec$ bird). Using facts, strict and defeasible rules, an agent is able to satisfy some literal h as in other rule-based systems. Let X be a set of facts in $\mathcal{L}$, STR a set of strict rules, and DEF a set of defeasible rules.

Given a set of ground atoms p_n, we introduce first *strong negation* $\sim$ to represent conflict between pieces of information p and $\sim p$. The set Lit of ground literals consists of ground atoms plus its $\sim$-negations; in addition, the *complement* function $\bar{\cdot} : \mathsf{Lit} \to \mathsf{Lit}$ assigns the conflicting literal $\overline{\ell}$ corresponding to each literal ℓ, i.e. $\overline{p} \longmapsto \sim p$ and $\overline{\sim p} \longmapsto p$.

In [García and Simari, 2004], the authors propose a non-monotonic consequence relation, called *warrant*, built upon the relation of defeat between constructible arguments for or against a literal. A defeasible logic program (or *de.l.p.*, henceforth) is a pair $T = (\Psi, \Delta)$ consisting of a strict and a defeasible part:

- a consistent set Ψ of *facts*, i.e. literals $\ell \in \mathsf{Lit}$ of the form p (for some $p \in \mathsf{Var}$) or its negation $\overline{p}$ (also, $\sim p$), where $\overline{\overline{\ell}} = \ell$, and,
- a set Δ of defeasible *rules*, where a defeasible rule is denoted as $\delta = \{\ell \prec \ell_0, \ldots, \ell_k\}$,

with rule $\ell \prec \ell_0, \ldots, \ell_k$ expressing that a warrant for $\ell_0, \ldots, \ell_n$ provide a (defeasible) reason for ℓ to be warranted. We denote $\mathsf{body}(\delta) = \{\ell_0, \ldots, \ell_n\}$ and $\mathsf{head}(\delta) = \ell$ representing the *body* and *head* of δ, respectively.

Derivability in T is closed under *modus ponens*: literals in Ψ are derivable and, given a rule δ, if each $\ell \in \mathsf{body}(\delta)$ is derivable, then $\mathsf{head}(\delta)$ is derivable. An argument for ℓ in a de.l.p. (Ψ, Δ), denoted $\langle \mathcal{A}, \ell \rangle$ or simply $\mathcal{A}$, is a set of rules $\mathcal{A} \subseteq \Delta$ such that:

- (i) ℓ is derivable from (Ψ, Δ),
- (ii) the set $\Psi \cup \mathcal{A}$ is non-contradictory, and,
- (iii) $\mathcal{A}$ is a minimal subset of Δ satisfying with respect to (i) and (ii).

Intuitively, an argument $\mathcal{A}$ is a minimal set of rules used to derive a conclusion. We denote $\mathsf{Rul}(\mathcal{A})$ as the set of rules of $\mathcal{A}$. The set of arguments constructible in (Ψ, Δ) is denoted $\mathsf{Args}(T)$. A derivation of a literal ℓ from (Ψ, Δ), still, does not suffice for its being warranted in (Ψ, Δ). This actually depends on the interaction among relevant arguments.

More specifically, the argument $\mathcal{A}$ for ℓ may be formed by:

$$\begin{aligned}
\mathsf{Rul}(\mathcal{A}) &= \{\delta_0, \delta_1\} \\
\mathsf{body}[\delta_0] &= \{p_0, p_1, q_2\} \\
\mathsf{body}[\delta_1] &= \{q_0, q_1, q_2\} \\
\mathsf{head}[\delta_0] &= \ell \\
\mathsf{head}[\delta_1] &= p_1 \\
\mathsf{concl}(\delta_0) &= \ell \\
\mathsf{concl}(\delta_1) &= p_1 \\
\mathsf{concl}(\mathcal{A}) &= \ell \\
\mathsf{base}(\mathcal{A}) &= \mathsf{body}[\mathsf{Rul}(\mathcal{A})] \setminus \mathsf{head}[\mathsf{Rul}(\mathcal{A})] \\
\mathsf{literals}(\mathcal{A}) &= \mathsf{body}[\mathsf{Rul}(\mathcal{A})] \cup \mathsf{head}[\mathsf{Rul}(\mathcal{A})]
\end{aligned}$$

In short, an argument $\mathcal{A}$ for ℓ, is a minimal non-contradictory set of defeasible rules, obtained from a defeasible derivation for a given literal ℓ. The literal ℓ will also be called the conclusion ($\mathsf{concl}(\mathcal{A})$) supported by $\mathcal{A}$. For instance figure 2.1 shows the proposed argument $\mathcal{A}$.

Figure 2.2 shows an instantiated example of a penguin, a chicken and a bird, and, shows a consistent set Ψ of *facts* and a set Δ of defeasible rules. The next arguments can be derived taking into account $T = (\Psi, \Delta)$ by an agent:

- $\langle \{\overline{flies(tina)} \prec chicken(tina)\}, \overline{flies(tina)} \rangle$
- $\langle \{flies(tina) \prec bird(tina)\}, flies(tina) \rangle$
- $\langle \{flies(tina) \prec chicken(tina), scared(tina)\}, flies(tina) \rangle$

The process to derive a conclusion of an argument $\mathcal{A}$ is as easy as going through the set of rules defeasible of the $\mathcal{A}$ from behind forward, or simply take the head of the last defeasible rule of the argument $\mathcal{A}$. Observe that in DeLP the construction of arguments is non-monotonic. That is, adding

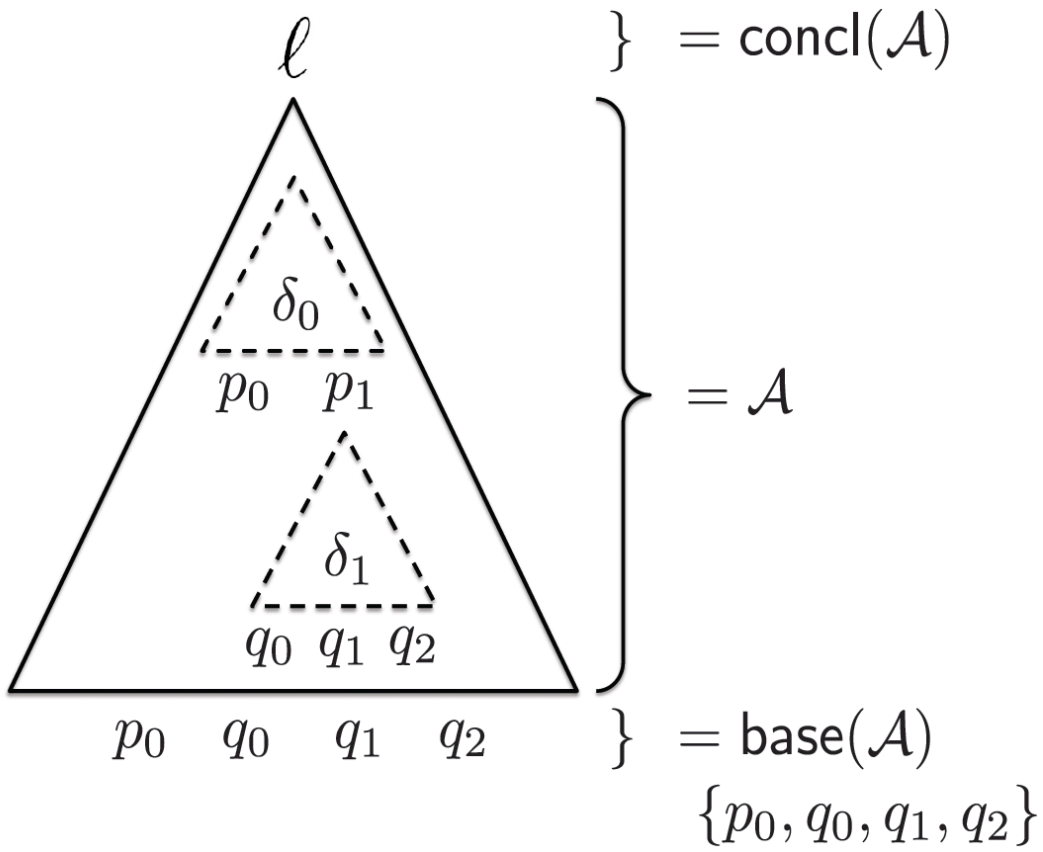

Figure 2.1: An argument $\mathcal{A}$ for ℓ made up of two rules δ_0, $\delta 1$.

facts or strict rules to the agent may cause some arguments to be invalidated because they become contradictory. More examples can be found in [García and Simari, 2004].

Given two arguments $\mathcal{A}, \mathcal{B}$, we say $\mathcal{A}$ *attacks* $\mathcal{B}$ if the conclusion of $\mathcal{A}$ contradicts some fact used in $\mathcal{B}$, that is, if $\overline{\mathsf{concl}(\mathcal{A})} \in \mathsf{literals}(\mathcal{B})$. This attack relation may roughly be seen as symmetric, in the sense that each attacked argument $\mathcal{B}$ contains a sub-argument $\mathcal{B}'$ attacking $\mathcal{A}$. (A *sub-argument* of $\mathcal{B}$ is a subset $\mathcal{B}' \subseteq \mathcal{B}$ supporting some inner conclusion ℓ' of $\mathcal{B}$, i.e. with $\ell' \in \mathsf{literals}(\mathcal{B})$.) To decide which contending argument prevails, a notion for preference among pairs of conflicting arguments is needed. A formal criterion for this lies in a comparison of information used in each argument: an attacking argument which makes use of more precise rules (or more information) is a *proper defeater* for -is preferred to- the contending argument. If two contending arguments are not comparable in these terms, they are a

$$\Pi = \left\{ \begin{array}{l} chicken(tina) \\ penguin(tweety) \\ scared(tina) \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} bird(X) \prec chicken(X) \\ bird(X) \prec penguin(X) \\ \sim flies(X) \prec penguin(X) \\ flies(X) \prec bird(X) \\ \sim flies(X) \prec chicken(X) \\ flies(X) \prec chicken(X), scared(X) \\ nests\_in\_trees(X) \prec flies(X) \end{array} \right\}$$

Figure 2.2: An instantiated example.

blocking defeater for each other[3].

Given an argument $\mathcal{A}_0$ for ℓ, an *argumentation line* $\Lambda = [\mathcal{A}_0, \ldots, \mathcal{A}_n]$ in (Ψ, Δ) is a sequence of arguments constructible in (Ψ, Δ), where each argument $\mathcal{A}_{k+1}$ is a defeater for its predecessor $\mathcal{A}_k$. Some further conditions are needed to rule out circular or inconsistent argumentation lines; briefly,

- arguments supporting (respectively interfering with) $\mathcal{A}_0$, i.e. of the form $\mathcal{B}_{2n}$ (respectively $\mathcal{B}_{2n+1}$) must form a consistent set, and
- no sub-argument $\mathcal{B}'$ of an argument $\mathcal{B}_{2m} \in \Lambda$ may appear later in Λ (i.e. it cannot be that $\mathcal{B}' = \mathcal{B}_{m'}$ with $m' > m$) see [García and Simari, 2004, García et al., 2008].

[3] Alternatively, one can specify by hand a preference between rules and then induce a defeat relation for arguments out of it.See [Simari and Loui, 1992] for details.

Since in a de.l.p. (Ψ, Δ) an argument can have several defeaters, different argumentation lines rooted in $\mathcal{A}_0$ can exist. This gives rise to a tree-like structure, the *dialectical tree* for $\mathcal{A}_0$, denoted $\mathcal{T}_{\mathcal{A}_0}(\Psi, \Delta)$. To check whether $\mathcal{A}_0$ is *defeated*, the following procedure on $\mathcal{T}_{\mathcal{A}_0}(\Psi, \Delta)$ is applied:

- label with a U (for *undefeated*) each terminal node in the tree (i.e. each non-defeated argument).

Then, in a bottom-up fashion, we label a node with:

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

A literal ℓ is *warranted* in (Ψ, Δ), denoted $\ell \in \mathsf{warr}(\Psi, \Delta)$, iff exists an argument $\mathcal{A}$ in (Ψ, Δ) with $\mathsf{concl}(\mathcal{A}) = \ell$ and $\mathcal{A}$ labeled U in $\mathcal{T}_{\mathcal{A}}(\Psi, \Delta)$.

In what follows we will specify more formally the set of introduced concepts of defeasible logic. In this way, we will detail a new definition for each new introduced concept. Note that this set of concepts represent the basis on which this thesis has been developed.

Definition 5. *A set* $\Psi \cup \Delta$ *is* non-contradictory *if no literal and its complement* $\ell, \overline{\ell} \in \{p_n, \sim p_n\}$ *can be derived from* $\Psi \cup \Delta$.

Briefly, a literal ℓ is warranted from (Ψ, Δ) if there is a non-defeated argument $\mathcal{A}$ supporting ℓ.

Definition 6. *Let* ℓ *be a literal, and* (Ψ, Δ) *a defeasible logic program.* $\langle \mathcal{A}, \ell \rangle$ *is an* argument *for* ℓ *if* $\mathcal{A} \subseteq \Delta$ *with*

(i) *there exists a derivation for* ℓ *from* $\Psi \cup \mathcal{A}$,
(ii) *the set* $\Psi \cup \mathcal{A}$ *is non-contradictory, and*
(iii) $\mathcal{A}$ *is* $\subseteq$*-minimal with respect to (i) and (ii)*

Comparison between conflicting arguments is induced from some comparison between rules occurring in them. This can be defined by (1) using some particular relation of preference[4] $> \subseteq \Delta \times \Delta$, or (2) using a a general formal method based on quantitative aspects of information occurring in arguments involved: given two conflicting arguments, the preferred argument the one that makes use of less rules (hence it is more specific), or makes use of more information (it is more precise). The suggested option in [García and Simari, 2004] is to combine these two criteria in a lexicographic

[4]This preference relation between rules will be based on the particular application domain.

way: we might use method (1) first, and if there is no preference, then apply method (2). In the present contribution, though, we opt for a more general choice relying on method (1) above.

In the examples below, we opt whenever possible for the general choice (1) above which captures the natural usage of defeasible logics. The formal definitions for these criteria are as follows. We consider first the general method (1):

Definition 7. *Let* (Ψ, Δ) *and* $\mathcal{F}$ *the set of literals derivable from* (Ψ, Δ)*. Let* $\langle \mathcal{A}, \ell \rangle$ *and* $\langle \mathcal{A}', \ell' \rangle$ *be arguments with* $\mathcal{A}, \mathcal{A}' \subseteq \Delta$*. We say* $\langle \mathcal{A}, \ell \rangle$ *is* strictly more specific than $\langle \mathcal{A}', \ell' \rangle$ *iff*

(1) For all $H \subseteq \mathcal{F}$*, if* ℓ *derives from* $H \cup \mathcal{A}$*, then* ℓ' *derives from* $H \cup \mathcal{A}'$*, and*

(2) For some $H' \subseteq \mathcal{F}$*,* ℓ' *derives from* $H' \cup \mathcal{A}'$ *but* ℓ *does not derive from* $H' \cup \mathcal{A}$*.*

If an argument $\langle \mathcal{A}, \ell \rangle$ *is more precise than another argument* $\langle \mathcal{A}', \ell' \rangle$ *then the former is* preferred*.*

For (2), the domain-dependent preference relation between arguments is induced from preference between rules as follows.

Definition 8. *Let* $>$ *be a priority relation between rules* $> \subseteq \Delta \times \Delta$ *(where* $\delta > \delta'$ *means* δ *is preferred to* δ'*). We define the following relation* $>$ *between arguments : an argument* $\langle \mathcal{A}, \ell \rangle$ *is* preferred over $\langle \mathcal{A}', \ell' \rangle$ *iff*

(i) for some $\delta \in \mathcal{A}$*,* $\delta' \in \mathcal{A}'$ *we have* $\delta > \delta'$*, and*

(ii) no element of $\mathcal{A}'$ *has priority over some element of* $\mathcal{A}$*.*

Having a notion of preference using the above definitions for (1) or (2), we are in position to define a relation of defeat (relating conflict with preference).

A fixed (Ψ, Δ) may give rise to conflicting arguments, e.g. arguments whose conclusions are in conflict (i.e. complementary). But it is not necessary that the conflict lies between arguments' conclusions: an argument may include a subargument whose conclusion conflicts with that of another argument.

Definition 9. *An argument* $\langle \mathcal{A}_1, \ell_1 \rangle$ attacks *or is a* counterargument *for* $\langle \mathcal{A}_2, \ell_2 \rangle$ *iff there exists a sub-argument* $\langle \mathcal{A}, \ell \rangle$ *of* $\langle \mathcal{A}_2, \ell_2 \rangle$ *such that* $\ell = \overline{\ell_1}$*. Argument* $\langle \mathcal{A}_1, \ell_1 \rangle$ *is a* proper defeater *for* $\langle \mathcal{A}_2, \ell_2 \rangle$ *iff* $\langle \mathcal{A}_1, \ell_1 \rangle$ *both attacks and is preferred to* $\langle \mathcal{A}_2, \ell_2 \rangle$*.*

This notion of defeat permits us to define argumentation lines:

Definition 10. *For a given* (Ψ, Δ) *and an argument* $\langle \mathcal{A}_0, \ell_0 \rangle$ *with* $\mathcal{A} \subseteq \Delta$*, an* argumentation line *for* $\langle \mathcal{A}_0, \ell_0 \rangle$ *is a sequence of arguments* $\Lambda = [\langle \mathcal{A}_0, \ell_0 \rangle, \langle \mathcal{A}_1, \ell_1 \rangle, \ldots, \langle \mathcal{A}_n, \ell_n \rangle]$ *such that each of its elements* $\langle \mathcal{A}_k, \ell_k \rangle$ *(for* $k > 0$*) is a defeater of its predecessor* $\langle \mathcal{A}_{k-1}, \ell_{k-1} \rangle$*.*

We say that subsequences of even elements $[\langle \mathcal{A}_0, \ell_0 \rangle, \langle \mathcal{A}_2, \ell_2 \rangle, \ldots]$ in an (acceptable) argumentation line Λ are the *supporting arguments* for $\langle \mathcal{A}_0, \ell_0 \rangle$, while the subsequence of odd arguments $[\langle \mathcal{A}_1, \ell_1 \rangle, \langle \mathcal{A}_3, \ell_3 \rangle, \ldots]$ in Λ are the *interfering arguments* for $\langle \mathcal{A}_0, \ell_0 \rangle$ in Λ.

In general, circular *counterargument* relations are problematic; this includes self-defeating arguments (whose conclusions are defeated by some of its sub-arguments), pairs of reciprocal arguments (each containing a subargument contradicting the other argument's conclusion), as well as subtler cases. The former case is ruled out in DeLP (see [García and Simari, 2004] Proposition 4.2). For the remaining cases, the next restriction is enough.

Definition 11. *(Adapted from [García and Simari, 2004, García et al., 2008]) An argumentation line*

$$\Lambda = [\langle \mathcal{A}_1, \ell_1 \rangle, \ldots, \langle \mathcal{A}_n, \ell_n \rangle]$$

is acceptable *iff*

(1) Λ *is finite, i.e.* $n \in \omega$

(2) The set of supporting arguments $[\langle \mathcal{A}_{2k+1}, \ell_{2k+1} \rangle]_{2k<n}$ *is consistent (that is,* $\bigcup_{2k<n} \mathcal{A}_{2k+1}$ *is consistent); similarly, for the set of interfering arguments* $[\langle \mathcal{A}_{2k}, \ell_{2k} \rangle]_{2k \leq n}$*.*

(3) No argument $\langle \mathcal{A}_k, \ell_k \rangle$ *in* Λ *is a subargument of an argument* $\langle \mathcal{A}_i, \ell_i \rangle$ *appearing earlier in* Λ *(i.e. with* $i < k$*), and*

(4) For all k *with* $1 < k < n$*,* $\langle \mathcal{A}_{k+1}, \ell_{k+1} \rangle$ *is a defeater of* $\langle \mathcal{A}_k, \ell_k \rangle$*.*

From the set of acceptable argumentation lines with root $\langle \mathcal{A}_1, \ell_1 \rangle$, one builds a dialectical tree $\mathcal{T}(\langle \mathcal{A}_1, \ell_1 \rangle)$, with the property that each node $\langle \mathcal{A}_{k+1}, \ell_{k+1} \rangle$ is a defeater of $\langle \mathcal{A}_k, \ell_k \rangle$. To decide whether ℓ is warranted from (Ψ, Δ), we label the nodes with a "U" (undefeated) or a "D" (defeated). We start with the leaves $\langle \mathcal{A}_n, \ell_n \rangle$ marked as U and then, in a bottom-up manner, we mark a node $\langle \mathcal{A}_k, \ell_k \rangle$ as U iff every child node has been marked as D (is defeated); otherwise (i.e. if some child is marked as U) $\langle \mathcal{A}_k, \ell_k \rangle$ is marked as D. Finally, we say that $(\langle \mathcal{A}, \ell \rangle)$ warrants ℓ (or ℓ is warranted, or $\ell \in \mathsf{warr}(\Psi, \Delta)$) if the root $\langle \mathcal{A}, \ell \rangle$ is marked as U. For instance figure 2.3 shows the warrant process for ℓ.

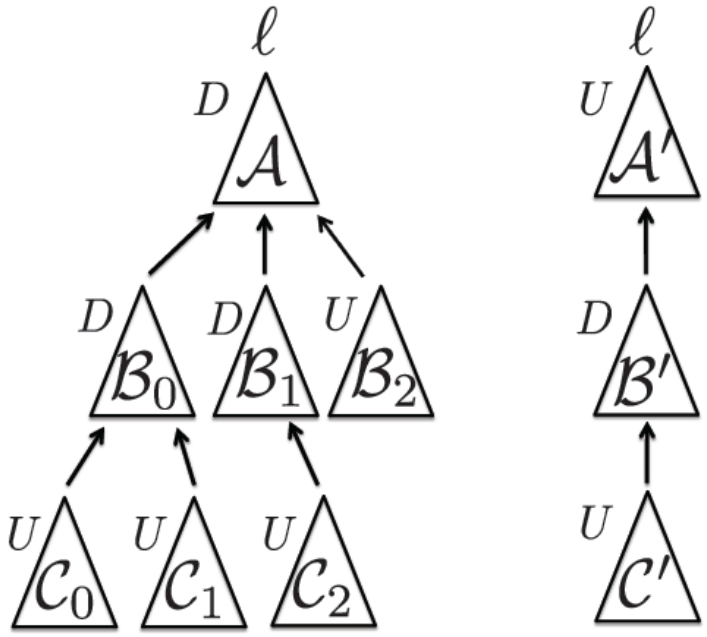

Figure 2.3: Computing warrant for l: an undefeated argument exists.

2.4 Argumentation in Planning

As we introduced in section 1.1, argumentation-based frameworks have been used in planning for reasoning about what actions are the best to be executed in a given situation. Argumentation has been applied on reasoning about conflicting plans and for generating consistent sets of intentions from a contradictory set of desires. Moreover, argumentation mechanisms allow agents to jointly devise a global shared plan and carry out collective actions and inferences.

2.4.1 A DeLP extension for POP planning

We recall here state-based and POP planning methods, before introducing DeLP-POP. In state-based planning, a solution is a linear sequence of actions, and thus before each action α_k, we know which state $\sigma_k \subseteq \mathsf{Lit}$ holds, with σ consistent.

In backward planning, the starting point is the set of goal G. Let g be

an open goal in G, and α an action that solves g; then, g is replaced in the set of open goals G by the set of preconditions of action α, i.e. $\mathsf{P}(\alpha)$.

Partial orderings also give rise to the notion of *threat*: an action step *potentially* interfering with some causal link. When detected, threats are to be solved by some threat resolution step. Thus in POP, the set of *flaws* to be solved in a plan Π includes threats and goals (the latter denoted by $G(\Pi)$, this set initially being G). Similarly, plan refinement steps refer in POP to ways to solve an open goal or an unsolved threat.

An extension of POP with DeLP-style argumentation, denoted DeLP-POP, was introduced in [García et al., 2008]. A DeLP-POP planner can appeal both to arguments and actions as a way to resolve goals or threats; so the original POP notions of planning domain, causal link and threat must be modified accordingly. As discussed at the beginning of Section 2.2, an *action* is a tuple of the form $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$, described by, respectively, sets of preconditions and effects. If literals in $\mathsf{P}(\alpha)$ are enforced (or warranted), then action α is *applicable* and its execution will enforce each $\ell \in \mathsf{X}(\alpha)$ (thus deleting $\overline{\ell}$ if holding previously). An argument $\mathcal{A}$ is *applicable* if $\mathsf{base}(\mathcal{A})$ is enforced; in which case $\mathsf{concl}(\mathcal{A})$ is derivable to support some precondition of α[5].

The notions of *link* and *threat* between action or argument steps of a POP become more complex in DeLP-POP. Let ℓ be an open goal, i.e. $\ell \in \mathsf{P}(\beta)$ for some $\beta \in A_\Pi$, or $\ell \in \mathsf{base}(\mathcal{A})$, for some argument $\mathcal{A} \subseteq \Delta$. If goal ℓ is planned to be enforced by an action α, this is encoded as a *causal link* of Π, in a set denoted $CL(\Pi)$: $(\alpha, \ell, \kappa) \in CL(\Pi) \subseteq A_\Pi \times G(\Pi) \times (A_\Pi \cup \mathcal{P}(\Delta))$, with $\kappa = \beta$ or $\kappa = \mathcal{A}$. If goal $\ell \in \mathsf{P}(\beta)$ is to be enforced by an argument, this is encoded as a *support link* of Π, in a set denoted $SL(\Pi)$: $(\mathcal{A}, \ell, \beta) \in SL(\Pi) \subseteq \mathcal{P}(\Delta) \times G(\Pi) \times A_\Pi$. Additional *ordering constraints* between action steps are encoded simply as $(\alpha, \beta) \in OC(\Pi) \subseteq A_\Pi \times A_\Pi$. If we abstract from the goals addressed in $CL(\Pi)$ and $SL(\Pi)$, the union of causal links, support links and ordering constraints $OC(\Pi)$ induce (by taking their transitive closure) the *partial order of* Π, i.e. the order between its steps, denoted $\prec$:

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

For a given agent, this search starts with an empty plan $\Pi^{(0)}$, only containing two dummy actions $\alpha_\Psi \prec \alpha_G$; these encode respectively Ψ as effects of α_Ψ, and G as preconditions of α_G (and otherwise empty). At each iteration, the algorithm nondeterministically selects an unsolved flaw and a

[5] See [García et al., 2008]'s backward planning algorithm for a full description of an *instance* of an action- or argument-steps, or an open goal *in a plan* Π. Each such instance κ is labeled by its full path of links up to some $g \in G$: $\langle \kappa, \ldots, g \rangle$.

refinement step for it (action-, argument- or threat resolution step); after this refinement, the algorithm updates the set of unsolved flaws, so goals and threats are added (if unsolved) or deleted (if solved).

Three kinds of threats must be checked during plan construction in DeLP-POP, see also Figure 2.4:

(a) action-action: $(\beta, (\alpha_0, \ell, \alpha_1)) \in A_\Pi \times CL(\Pi)$, s.t. $\overline{\ell} \in \mathsf{X}(\beta)$; here β threatens the link between α_0 and α_1

(b) action-argument: $((\beta, \overline{n}), (\mathcal{B}, b, \alpha_1)) \in (A_\Pi \times \mathsf{Lit}) \times SL(\Pi)$, with $\overline{\mathsf{X}(\beta)} \cap \mathsf{literals}(\mathcal{B}) \supseteq \{n\}$, where β contradicts some step used in $\mathcal{B}$ to derive b, and

(c) argument-argument $((\beta, c_k, \mathcal{C}), (\mathcal{B}, b, \alpha_1)) \in (A_\Pi \times \mathsf{Lit} \times \mathcal{P}(\Delta) \times SL(\Pi)$, with $\mathcal{C}$ defeating $\mathcal{B}$.

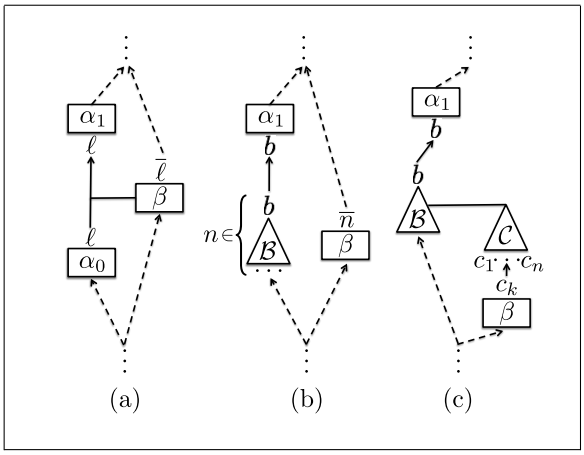

Figure 2.4: Threat types: (a) action-action, (b) action-argument and (c) argument-argument.

Different maneuvers, consisting in new ordering constraints for the threat, and/or new plan steps against it, may be tried to resolve each kind of threat. These maneuvers are inspired by threat resolution steps in standard POP. See Figure 2.5 for resolution steps to action-action threats (a') and (a"), and action-argument threats (b') and (b"). See Figure 2.6 for resolutions steps to argument-argument threats. Note that in (c") and (c"') action α_1 must be modified with a new precondition $\overline{p}$; this allows to create a link between $\mathcal{A}$ and α_1 in the resolution step.

We refer the reader to the algorithm described in [García et al., 2008] for a more detailed account.

Under this new perspective, we reformulate the definition 1 as follows:

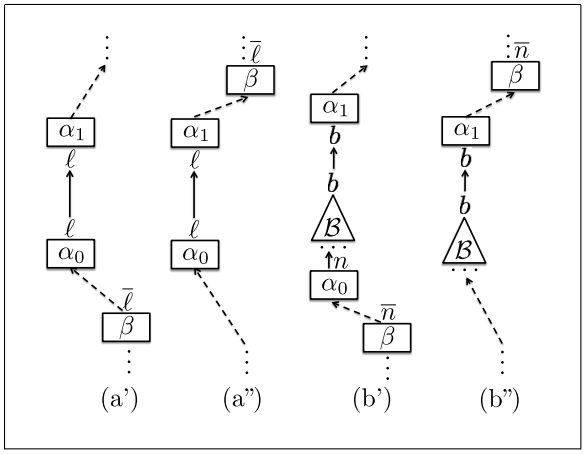

Figure 2.5: Solutions to (a), (b). Promote: (a'), (b'); and Demote: (a"), (b").

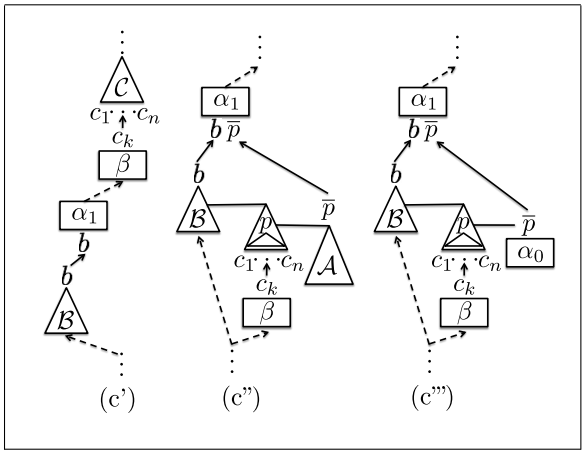

Figure 2.6: Solutions to (c): Delay (c'), Defeat (c") and Disable (c"').

Definition 12. *A **partial-order plan** is a tuple* $\Pi = \langle A_\Pi \cup AS_\Pi, OC, CL \cup SL, OC, \mathsf{Threats} \rangle$*, where* A_Π*,* OC*,* CL *and* Threats *have the usual meaning,* AS_Π *is the set of argument steps included in* Π *and* SL *is the set of support links.*

3

Multi-agent DeLP-POP: a new general framework based on Cooperative Planning

In this section, we survey the main problems a multi-agent extension of DeLP-POP must face and present an extension of the single-agent framework to cope with these problems. The novelty of our work is the combination of the next aspects:

- Defeasible reasoning,
- Decentralized planning, and,
- Multi-agent systems.

3.1 Concepts for a multi-agent DeLP-POP

The main modifications we have performed to extend a single-agent DeLP-POP to a multi-agent scenario are:

1. a process for a *collaborative* detection of existing threats,
2. a process for a *collaborative* proposal of plans, and,
3. a process for a *collaborative* construction of arguments.

Let us to introduce a set of concepts that we have implemented with the objective of addressing the multi-agent DeLP-POP framework. The following concepts are modifications needed for extending the single-agent framework [García et al., 2008] presented in section 2.4.1.

3.1.1 The planning domain of the agents

An agent $x \in \mathsf{Ag}$ from our multi-agent DeLP-POP framework is initially endowed with a planning domain $\mathbb{M}_x$. A planning domain is a tuple $\mathbb{M}_x = (T_x, A_x, G)$ where $T_x \subseteq \mathsf{Lit}$ represents (beliefs about) the initial world, A_x is a set of actions and $G \subseteq \mathsf{Lit}$ is the set of goals of the agent. A solution is a plan Π composed of a set of actions A_Π leading a T-world into a G-world by means of the actions in the plan $A_\Pi \subseteq A_x$. Instead of T_x being just a set of literals, T_x is now defined as a tuple (Ψ_x, Δ_x) where:

- Ψ_x contains the literals that are *initially* true in the initial world, i.e., facts or indisputable statements, and,
- Δ_x contains the defeasible rules that may apply anywhere in the plan.

Summarizing, the planning domain of an agent is composed of the following components:

- Ψ_x: a consistent set of initial facts (literals),
- Δ_x: a set of defeasible rules,
- A_x:, a set of actions, and,
- G: a set of goals.

Note that we do not consider private goals in this first approach of a multi-agent DeLP-POP framework. Therefore, multi-agent DeLP-POP is not able to work with non-cooperative agents. In fact, we will assume that all agents are fully cooperative and they have only a set of common goals. ConsideringThe study the choice of having non-cooperative agent is mentioned as a future work of this master thesis project (see section 6.2).

3.1.2 Proto-states

A partial order plan (henceforth: plan) Π is a set of actions whose execution ordering (i.e. links on action pairs) is only *partially specified*, thus encoding multiple linear plans. Hence, instead of states, a plan Π determines a (possibly inconsistent) set of facts that are potentially achievable before action α. This set, which here is called the *proto-state* of α, is denoted as S^{Π}_{α}. Informally, a proto-state S^{Π}_{α} is referred to be the set of facts which could be true before the action α.

Threat detection is based on the concept of *proto-states* in our multi-agent DeLP-POP model, defined next. More formally, for a fixed $\mathbb{M} = ((\Psi, \Delta), A, G)$, a plan Π and $\alpha \in A_{\Pi}$, the proto-state S^{Π}_{α} denotes the set of literals that can be potentially true before executing α, if we extend $\prec_{\Pi}$ with an arbitrary new constraint[1]:

$$
\begin{aligned}
S^{\Pi}_{\alpha} = \{\ell \in \mathsf{Lit} \mid \exists \alpha' \in A_{\Pi} \text{ s.t. } \ell \in \mathsf{X}(\alpha') \text{ and } \prec_{\Pi} \cup \{\langle \alpha', \alpha \rangle\} \\
\text{is consistent, and } \forall \beta \in A_{\Pi}, \text{ if } \overline{\ell} \in \mathsf{X}(\beta) \text{ then} \\
\{\langle \alpha', \beta \rangle, \langle \beta, \alpha \rangle\} \not\subseteq \mathsf{tc}(\prec_{\Pi} \cup \{\langle \alpha', \alpha \rangle\})\}
\end{aligned}
$$

We use the proto-state S^{Π}_{α} to compute which arguments $\mathcal{A} \subseteq \Delta$ might unintentionally be triggered in the Π.

An argument $\mathcal{A}$ is *applicable* at S^{Π}_{α} (see Figure 3.1) if $\mathsf{base}(\mathcal{A})$ is enforced in S^{Π}_{α}; in which case $\mathsf{concl}(\mathcal{A})$ is derivable and it may support some precondition of α.

In what follows, we explain an example which shows the content of a proto-state at different times during the search in the space of plans. The example allows us to understand how evolution in the construction of the plan Π implies an updating process in the proto-states. On the one hand, if we focus our attention on the action α on the left side in Figure 3.2, we can distinguish two types of actions in the plan Π:

- Actions that must necessarily occur before α: α' and α'', and,
- Actions that may occur before α: β and $\beta*$.

Thus, the set of facts which could be true before the action α (proto-state S^{Π}_{α}) is $S^{\Pi}_{\alpha} = \{\overline{p}, q, \overline{q}\}$. Note that α'' is ordered before α, and α'' deletes p, so $p \notin S^{\Pi}_{\alpha}$.

On the other hand, if we focus our attention on the action α on the right side in Figure 3.2, we can distinguish two types of actions in the plan Π':

[1] Note that S^{Π}_{α} is computed as if α were actually applicable. In particular, arguments occurring before α play no role for S^{Π}_{α}.

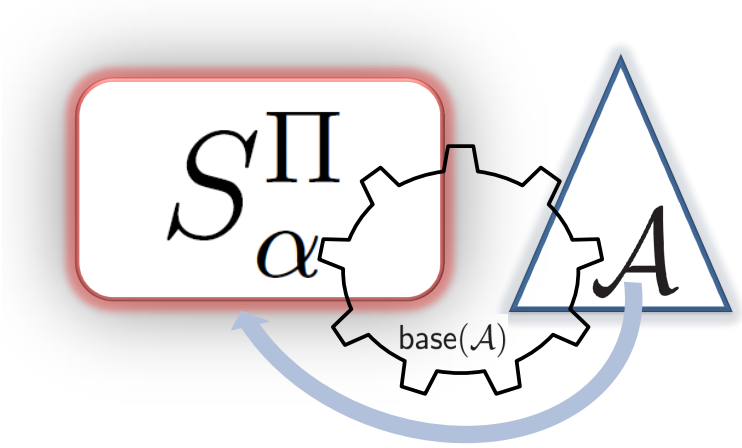

Figure 3.1: An argument $\mathcal{A}$ applicable at the proto-state S^{Π}_{α}.

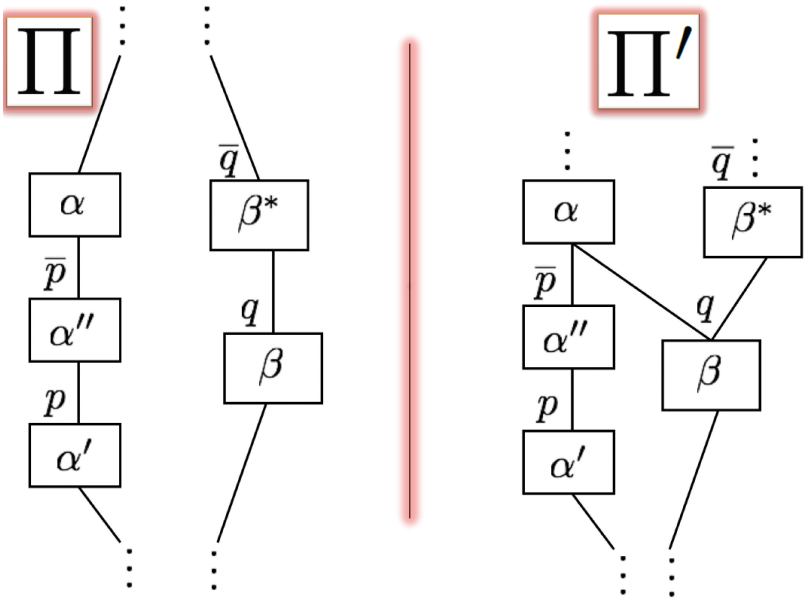

Figure 3.2: How calculate the proto-states in the construction of the different plans.

- Actions that must necessarily occur before α: α', α'' and β, and,
- Actions that may occur before α: $\beta*$.

Although the plan Π has changed to Π', the proto-state S^{Π}_{α} does not

change. However, if $\beta*$ were ordered before α, S^{Π}_{α} would change.

3.1.3 Pre-arguments

Multi-agent DeLP-POP introduces the concept of *pre-argument* $\mathcal{A}$. A *pre-argument* $\mathcal{A}$ is a tuple of literals and rules $(p\mathsf{base}(\mathcal{A}), \mathcal{A})$, where $p\mathsf{base}(\mathcal{A})$ contains all literals from $\mathsf{base}(\mathcal{A})$ but some of them are underlined: $\underline{\ell} \in p\mathsf{base}(\mathcal{A})$, which means that:

1. the agent does not know that ℓ holds in the corresponding proto-state, and,
2. the agent cannot infer the literal through the actions so it will attempt to support the literal with an argument step.

Otherwise, if ℓ occurs non-underlined, we say $\ell \in \mathsf{base}^{+}(\mathcal{A})$. We denote the set of pre-arguments in a proto-state S^{Π}_{α} as $\mathsf{PArgs}(S^{\Pi}_{\alpha}, \Delta) := \{(X, \mathcal{A}) \mid X \subseteq S^{\Pi}_{\alpha}, \mathcal{A} \subseteq \Delta$ and for each $\ell \in \mathsf{base}(\mathcal{A})$, $X(\ell) \in \{\ell, \underline{\ell}\}\}$.

Informally, a pre-argument is an argument with a partial knowledge base which is to be supplemented by:

- the facts known by other agents, and,
- the conclusions derived through the defeasible rules of other agents.

For example, in Figure 3.3, $\mathcal{A}$ is a pre-argument proposed by some agent $x_1 \in \mathsf{Ag}$, and $\mathcal{B}$ is an argument proposed by some other agent $x_2 \in \mathsf{Ag}$, which appears to complete $\mathcal{A}$. It is the set of literals (also known as in the set of underlined literals) base of an argument proposed by an agent, and that the agent cannot solve by himself, i.e. the agent has not the sufficient knowledge (actions) to deduce such a literal. For this reason, other agents are allowed to help the agent support the literal through the use of arguments. Thus, a collaborative argument construction process is implemented by the concept of pre-argument in multi-agent DeLP-POP. Figure 3.4 shows this example executed between the agent x_1 and x_2.

3.1.4 Cost of the actions and inferences

We introduce the *cost* of an action, e.g. define action α as $\langle \mathsf{P}(\alpha), \mathsf{X}(\alpha), \mathsf{cost}(\alpha)\rangle$ where $\mathsf{cost}(\alpha) \in \mathbb{R}$. Given two plans Π_0 and Π_1, we say Π_0 is cost-preferred to Π_1 iff $\mathrm{cost}(\Pi_0) < \mathrm{cost}(\Pi_1)$, where $\mathrm{cost}(\Pi)$ is the aggregated cost function of Π_1. We will consider $cost(\Pi) = \sum_{\alpha \in \Pi} cost(\alpha)$.

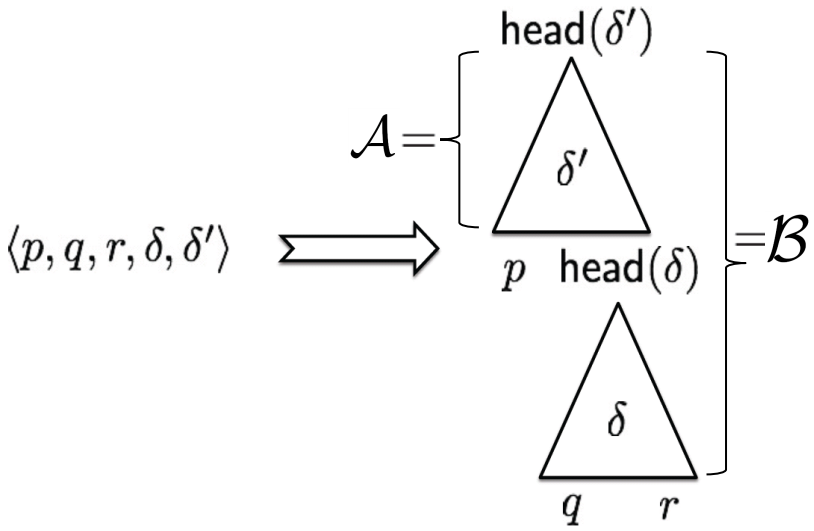

Figure 3.3: An argument $\mathcal{B}$ refining an pre-argument $\mathcal{A}$ (I).

As we have previously introduced, an open condition of the plan can be satisfied by means of an action or an inferences process about the environment. Given a non-defeated argument $\mathcal{A}$ inferring a open goal ℓ, we consider that the *cost* of the argument $\mathcal{A}$ as $\mathsf{cost}(\mathcal{A}) = 0$.

3.1.5 Agent's learning ability

Communication of facts, rules or actions from agent x to an agent y will be assumed as an expansion with respect to Ψ_y, Δ_y, and A_y of $\mathbb{M}_y$.

This aspect allows agents in a multi-agent DeLP-POP to learn (see Figure 3.5) and improve their beliefs and defeasible rules based on dialogues (see sections 3.3.2 and 3.3.1) with other agents. However, we do not consider a specific learning model [Carmel and Markovitch, 1996] in this master thesis project, thus learning is as easy as incorporating to the planning model which new information from a dialogue that not has been previously considered by an agent.

3.1.6 Absolute and non-absolute threats

As we detail in section 2.4.1, three kinds of threats must be checked during plan construction in DeLP-POP:

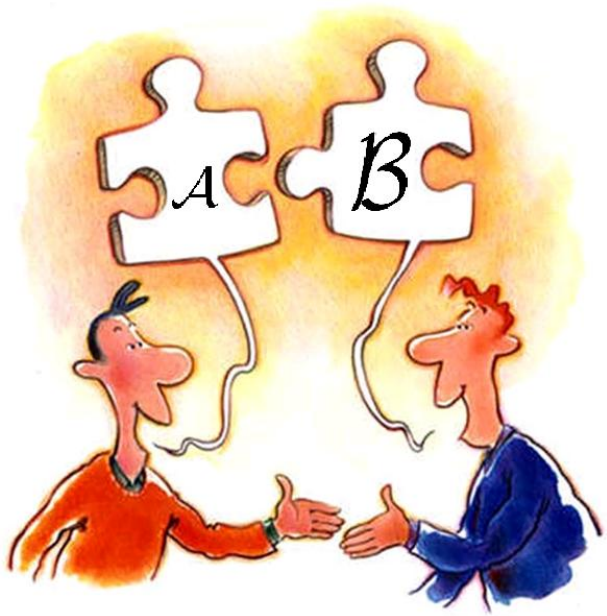

Figure 3.4: An argument $\mathcal{B}$ refining an pre-argument $\mathcal{A}$ (II).

1. action-action,
2. action-argument, and,
3. argument-argument.

We keep these three kinds of threats in a multi-agent DeLP-POP. We sustain that the fact of considering more agents during the plan search does not increase these three kinds of threats. However, it is important to disntinguish between absolute threats and not absolute threats.

- Absolute threats: This is the set of threats which do not change under an increase in the number of agents who are part of a dialogue phase; in other words, these are the threats that always remain in the plan no matter the number of agents neither the defeasible knowledge of the agents. (see sections 3.3.2 and 3.3.1):
  - action-action, and
  - action-argument.

Figure 3.5: Agents improving their beliefs and defeasible rules by means of dialogues.

- Non-absolute threats: This is the set of threats that can change according to the number of agents who are part of a dialogue phase.

  – argument-argument.

The sets of action-action threats and action-argument threats are absolute under expansions of T. In contrast, expanding Ψ or Δ (i.e. by learning new facts or rules) may activate a new argument that defeats (the derivation of) a literal previously thought to hold. Moreover, such a new arguments might constitute new argument-argument threat to some step in $SL(\Pi)$.

Therefore only some class of threats (argument-argument) is not absolute, and hence discussion between agents for this type of threats makes sense (see section 3.3.1). An argumentative exchange will suffice for agents to agree on the presence of threats. On the other hand, for other types of threats, discussion would not increase the set of detected threats: they can be discovered by each agent with the only help of the knowledge implicitly communicated with the plan itself.

Following the same idea as before, the proto-states S^{Π}_{α} are absolute under expansions of T; i.e., the proto-state S^{Π}_{α} is not changed by the expansion of T, but by the evolution of the plan Π.

3.2 Overview of Cooperative Planning

The previous section introduced the general concepts of the multi-agent DeLP-POP. Now, we present an overview of cooperative planning algorithm. In this section, we detail the design and specification of a DeLP-POP framework as an extension of the traditional POP algorithm by considering, among other things, the introduction of argument steps and corresponding support links, and multiple agents involved in the discovering of new threats.

The traditional POP algorithm works as follows: starting with the initial empty plan $\Pi^{(0)}$ (step 1 in Algorithm 1), it works through the application of successive refinement steps at each iteration. First, it chooses a partial-order plan from the list of candidates (step 3 in Algorithm 1), and then it applies a refinement step that involves selecting a flaw (threat or open condition) in the partial-order plan or it evaluates the selected plan.

The two non-determinist **choose** steps (see step 3 and step 15 in the algorithm 1) determine that the algorithm has to make a choice among different alternatives:

- Selecting the next partial-plan to work on (the next refinement plan solving it), and,
- Selecting the next open condition/threat to study in the partial plan, i.e the selection of next flaw to solve in the partial plan chosen in the above step.

Typically, the selected choice will be the result of the application of a specific heuristic [Ghallab et al., 2004] for selecting of plans, and a different heuristic to select the next flaw.

3.2.1 Extending the single-agent DelP-POP to multi-agent DeLP-POP

Algorithm 1 shows an outline of the multi-agent DeLP-POP model which has been developed in this master thesis project. On the one hand, in contrast with the traditional POP algorithm, algorithm 1 considers argument steps, besides action steps, to support unsatisfied preconditions (flaws). On the other hand, in contrast with Simari's work [García et al., 2008], algorithm 1 considers multiple agents involved in the detection of new threats, and multiple agents dialoguing about new refinement proposals.

In the following, we assume we have a set of agents $\{x_0, x_1, ...\} \in \mathsf{Ag}$, each one with a planing domain $\mathbb{M}_x = ((\Psi_x, \Delta_x), A_x, G_x)$ (see section 3.1.1).

In purely cooperative scenarios, agents have no individual interests (i.e. $G_{x_0} = G_{x_1}$ for any $x_0, x_1 \in \mathsf{Ag}$) and hence no incentives to retain relevant information. Moreover, we assume $\bigcup_{i \in \mathsf{Ag}} \Psi_i$ is a consistent set. Multi-agent DeLP-POP takes $\bigcup_{x=0}^{n} \mathbb{M}_x$, where n is the amount of agents to cooperate in the planning search.

$\mathsf{flaws}(\Pi) = G(\Pi) \cup \mathsf{Threats}(\Pi)$ in step 3 represents the set of open goals, and threats of the plan Π, respectively; $\mathsf{Threats}(\Pi)$ addresses the three types of threats that can appear in a plan and that we mentioned in section 2.4.1.

A plan Π is added to the set SolutionPlans in step 5, if Π has not unsolved flaws. Note that the proposed algorithm 1 does not end with the first solution found, but it searches the full space of solutions. Thereby, Algorithm 1 is guaranteed to find an *optimal solution* (if there is a solution), regardless of the choice heuristics used in step 3 and step 15.

We can greatly simplify multi-agent planning, at least in cooperative scenarios, by using a dialogue-based plan search procedure. On the one hand, steps from 8 to 13 in algorithm 1[2] show a dialogue-based plan evaluation phase where multiple agents are involved in the detection of new threats (see section 3.3.1). On the other hand, steps from 15 to 21 in algorithm 1 show a dialogue-based plan search phase where new refinement plans are proposed to continue the joint process of building a plan that solves G (see section 3.3.2). The process ends when the set Plans is empty, in whose case SolutionPlans is the set of solution plans.

To sum up, two kinds of dialogue are clearly differentiated in the algorithm 1:

1. Dialogue-based plan evaluation, and,

2. Dialogue-based plan search.

The main novelty of these two dialogues is that such a dialoguing group of planner agents actually implements a search procedure. In the next sections, we will explain in more detail each step of the algorithm 1.

3.3 Dialogues for multi-agent DeLP-POP

The previous section introduced an overview of cooperative planning and an outline of the multi-agent DeLP-POP algorithm. Now, we present the two

[2] (n, i) denotes the turn to the agent i in the iteration n in Algorithm 1.

1: Plans := $\Pi^{(0)} := \{\alpha_0 \prec \alpha_G\}$
n:=0
2: **while** Plans $\neq \emptyset$ **do**
3: **choose** $\Pi \in$ Plans
flaws(Π) = $G(\Pi) \cup$ Threats(Π)
4: **if** (flaws(Π) = $\emptyset$) **then**
5: SolutionPlans := SolutionPlans $\bigcup \Pi$
go to 3
6: **else**
7: MultiThreats := 0
8: **for each** $i \mid (i \in$ Ag) **do**
9: MultiThreats := MultiThreats $\bigcup \{\Pi^{(n,i)}\}, \forall\ \Pi^{(n,i)}$ that refines Π {*Each* $\Pi^{(n,i)}$ *is a new 'derived plan' including the threats envisaged by the rest of agents*}
10: **for each** $x \mid ((x \in$ Ag) & $(x \neq i))$ **do**
11: Update ψ_x, Update Δ_x
12: **if** (MultiThreats $\neq \emptyset$) **then**
13: Plans := Plans $\bigcup$ MultiThreats
go to 3
14: **else**
15: **choose** $\Phi \in$ flaws(Π)
16: **for each** $i \mid (i \in$ Ag) **do**
17: *Relevant* := *Relevant* $\bigcup \{\Pi_r^{(n,i)}\}, \forall\ \Pi_r^{(n,i)}$ that resolves Φ {*Each r is a choice (partial-order planning) to solve* Φ}
18: **for each** $x \mid ((x \in$ Ag) & $(x \neq i))$ **do**
19: Update ψ_x, Update Δ_x, Update A_x
20: **if** *Relevant* $\neq \emptyset$ **then**
21: Plans := Plans $\bigcup$ *Relevant*
$n \leftarrow n + 1$
22: **if** (SolutionPlans = $\emptyset$) **then**
23: return fail {*Not exists plan*}
24: **else**
25: return SolutionPlans

Algorithm 1: Outline of the multi-agent DeLP-POP algorithm

types of dialogues, for multi-agent plan evaluation and plan search. Corresponding formal results of agreement among agents after these dialogues terminate are also shown.

More precisely, the purpose of the multi-agent argumentative dialogues is to let agents:

- Agree upon evaluation of a plan by means of an argument-argument threat detection. This kind of threats are labeled as non-absolute threats (see section 3.1.6), and for this reason they are detected through a dialogue process between agents as we will see in Section 3.3.1.
- Support decentralized plan search, allowing agents to refine or revise plans or to persuade other agents to accept a plan as we will see in Section 3.3.2.

3.3.1 Dialogue-based plan evaluation

This section introduces a first simple dialogue (from steps 7 to 14 in the algorithm 1) to evaluate a fixed plan. Roughly, the problem stems from different agents discussing about a particular given plan; since these agents may have different beliefs (initial facts, rules) they need not agree on the evaluation of the plan at some step. This evaluation dialogue will conduct the resolution of future threats.

Algorithm 1 presents a turn-based dialogue (an agent talking only during her turns) from step 8 to 11, allowing agents $\{x_0, x_1\} \in \mathsf{Ag}$ to collaborate to discover threats to a fixed argument step $\mathcal{A}$ of $SL(\Pi)$, where Π is a selected plan in step 3 in algorithm 1. Both agents may contribute to argue pro and against $\mathcal{A}$ in S^{Π}_{α}.

The input to this dialogue is the selected plan Π in step 3 in Algorithm 1. Basically, the dialogue consist of discovering new argument-argument threats based on the facts and defeasible rules of the agents. In this dialogue, each agent, at his turn, informs about the threat discovered in the plan, and this dialogue ends when all agents have received their turn. Note that detecting threats in the plan Π simply consists in sending an argument $\mathcal{B}$ which defeats some argument $\mathcal{A}$ which supports the precondition of some action in Π.

At each turn n, an agent i sends a set of pre-arguments which defeat the argument $\mathcal{A}$ used as a support link in the plan Π. For each detected threat the agents has to derive the selected plan Π to a plan $\Pi^{(n,i)}$, where $\Pi^{(n,i)}$ is simply Π with the detected argument-argument threat by the agent (see section 2.4.1):

- n represents the current iteration in Algorithm 1, and,

- i represents the agent.

We use the following graphical representation to denote arguments and actions:

- Arguments are depicted as triangles. The upper vertex of the triangle is labeled with the argument's conclusion, and the bottom of the triangle is labeled with the argument's base. An argument is identified by a letter of $\{\mathcal{A}^x, \mathcal{B}^x, \mathcal{C}^x, ...\}$, where the superscript x indicates the agent who proposed the argument.

- Actions are depicted as rectangles. The top of the rectangle is labeled with the effects of the action α, i.e., $\mathsf{X}(\alpha)$, and the bottom of the triangle is labeled with the preconditions of the action α, i.e., $\mathsf{P}(\alpha)$. An action α is identified by a short name. For example, the short name mP denotes 'moving plane'.

For instance, Figure 3.6 shows a selected plan Π in step 3 in Algorithm 1 as the input to this dialogue, and Figure 3.7 shows a derived plan $\Pi^{(n,i)}$ including the discovered argument-argument threat (by the agent Ann) as the output of this dialogue.

As shown in step 10 and step 11, for each proposed pre-argument $\mathcal{B}$, the other agents, learn as initial facts those literals stated in $\mathcal{B}$ that are not in the their view of the proto-state, i.e. with $\ell \in \mathsf{base}^{+}(\mathcal{B})$ and $\ell \in S^{\alpha}$.

Informally, the agents learn (see Section 3.1.5) those literals from each proto-state which are not known by them. Hence, so-learned literals ℓ must have come from the other agent's ψ-set and propagated to this proto-state. All rules from $\mathcal{B}$ which are novel to the agent are learned as well.

The output of this dialogue is a set of plans $\mathsf{MultiThreats}$ where each $\Pi^{(n,i)} \in \mathsf{MultiThreats}$ derives Π. In multi-agent DeLP-POP, two types of outputs from this dialogue are differentiated:

- $\mathsf{MultiThreats} = \emptyset$: there is not any detected threat to the plan Π by the agents, and then Algorithm 1 jumps to step 15 to start the dialogue-based search in the space of plans.

- $\mathsf{MultiThreats} \neq \emptyset$: at least there is one detected threat by the agents, the set of new plans $\mathsf{MultiThreats}$ is added to Plans, and the Algorithm 1 jumps to step 3.

Figure 3.8 intends to be a scene that shows graphically the activity carried out by a set of five agents ($\{x_0, x_1, x_2, x_3, x_4\} \in \mathsf{Ag}$) during the dialogue-based

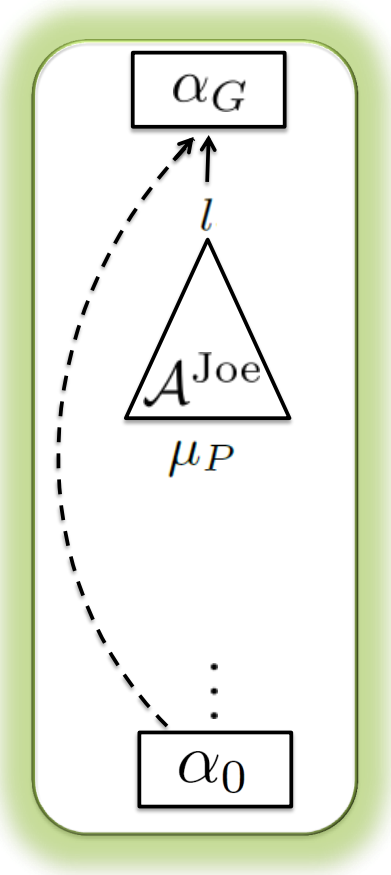

Figure 3.6: A selected plan Π.

plan evaluation. A plan Π is selected by step 3 of algorithm 1, and a set of agents start an argumentative discussion to evaluate the feasibility of the plan.

Since the the set of facts ψ and defeasible rules Δ of each agent is finite, the dialogue-based plan evaluation will always terminate in a finite number of iterations (see step 8 in the algorithm 1). Note that one evaluation dialogue is required for each selected plan.

3.3.2 Dialogue-based search in the space of plans

This section introduces a second dialogue (from steps 16 to 21 in the Algorithm 1) to propose different refinement plans which support the selected flaw Φ, such that $\Phi \in \mathsf{flaws}(\Pi)$. When we say *flaw* we refer to both an open

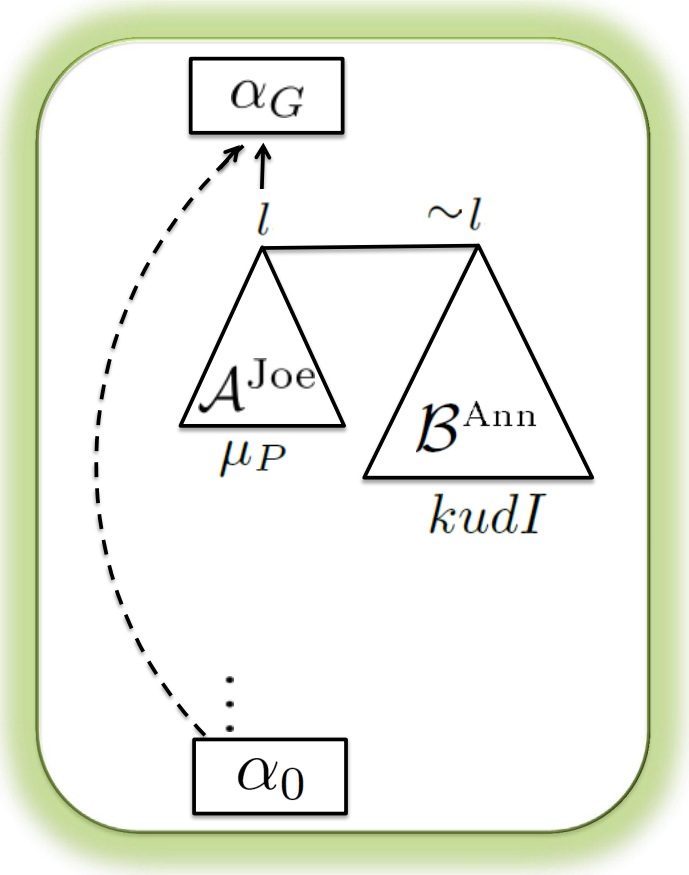

Figure 3.7: A derived plan $\Pi^{(n,Ann)}$.

goal or a threat in the plan Π. Each refinement, indicated as 'r' in Algorithm 1, gives rise to a refinement plan that is included in the set '*Relevant*'.

Multi-agent DeLP-POP works on a planning process distributed among several planning/executing agents who devise a joint, non-linear plan which be later executed by the same agents. We assume that agents are specifically designed to be cooperative, so the agent's decisions must only be derivative from the collective goals G. Domain knowledge is usually distributed among agents, so agents typically works with an incompletely known domain, i.e., the set of actions that an agent can propose is different from other agents'.

The overall goal of this dialogue is to enable agents to propose different plans. The main difference with respect to traditional POP planners is that DeLP-POP uses two ways to support the flaw Φ:

- Supporting the flaw Φ with an action step: the semantics is the same

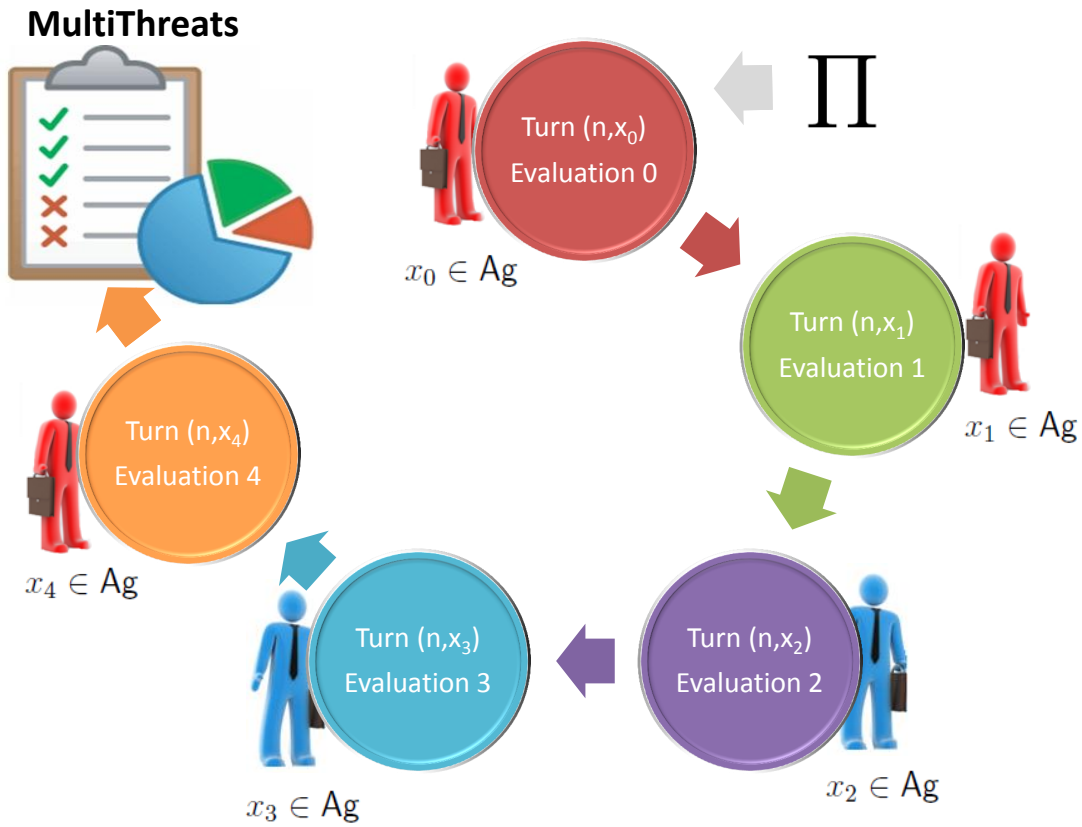

Figure 3.8: Evaluating a selected plan Π.

as in classical planning; a precondition Φ is achievable by some action $\alpha \in A$, or the threat is solved by imposing an ordering constraint between two actions, and,

- Supporting the flaw Φ with an argument step: Φ is derivable by some proposed argument $\mathcal{A}$.

Thus, a plan Π can be refined with an argument step or an action step.

This dialogue refers to multiple agents planning and acting also collaboratively. More specifically, agents interact to design a devised plan in algorithm 1 that none of them could have generated individually in most cases. This dialogue during the plan construction is also (see section 3.3.1) realized as a turn-based dialogue between the agents. The agents keep in mind that the devised plan will be jointly executed by themselves such that they collec-

tively achieve the common goals G (G is known by all of the agents) of the problem.

Following, we explain the steps involved in this dialogue:

1. The input to this dialogue is the selected plan Π in step 3 in Algorithm 1 and the selected flaw Φ according to some heuristic function [Gerevini and Schubert, 1996, Geffner, 2005, Penberthy and Weld, 1992b] in step 15 in Algorithm 1.

   each agent, at his turn, informs about the threat discovered in the plan, and

2. The turns round (steps from 16 to 21) between the agents start in the same way that the previous dialogue-based plan evaluation (see section 3.3.1). In this dialogue, each agent, at his turn, proposes proposals as alternatives to achieve[3] or derive[4] the selected flaw Φ, and this dialogue ends when all agents have had a turn. The set of proposals (by the agent) which refine the selected plan Π are also labeled as $\Pi_r^{(n,i)}$, where n also represents the current iteration in Algorithm 1, i represents the agent, and r represents the refinement proposal by the agent.

   At each turn i, an agent can propose as many plans as possible from their knowledge.

   - To update one's set of initial facts, an agent will, as before, extract literals from (pre-)arguments' bases, but also will learn those literals ℓ in received plan proposals. Unlike the previous dialog where the agent's learning ability is focused exclusively on the facts or literals and defeasible rules, here multi-agent DeLP-POP introduces also update of non-initial actions, i.e., this actions which were not known at the beginning of the problem by the agent.

3. The output of this dialogue is a set of plans *Relevant* where each $\Pi_r^{(n,i)} \in$ *Relevant* extends Π. In multi-agent DeLP-POP, two types of outputs for this dialogue are also differentiated:

   - *Relevant* $= \emptyset$: there is not any proposal to resolve the flaw Φ in Π by the agents. The algorithm 1 implicitly prunes Π due to this reason that the selected plan Π has not been refined by the agents, and then the algorithm jumps to step 2.

[3] action steps and causal links

[4] argument steps and support links

- *Relevant* $\neq \emptyset$: at least there is refinement proposed by the agents, the set of new plans *Relevant* is added to Plans, and the Algorithm 1 jumps to step 3.

For instance, Figure 3.9 shows a selected plan Π in step 3 in Algorithm 1 as the input to this dialogue, where $\Phi = \mu_p$, and Figure 3.10 shows a refinement plan $\Pi_r^{(n,i)}$ to Π as the output of this dialogue.

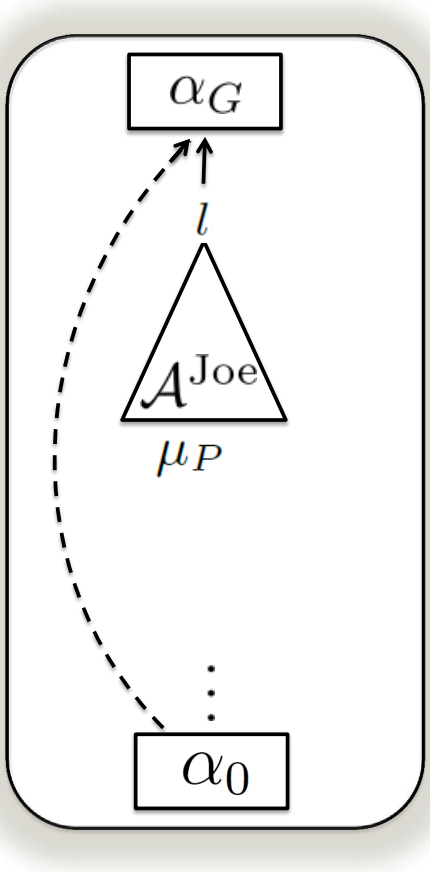

Figure 3.9: A selected plan Π.

Figure 3.11[5] intends to be a scene that shows graphically the activity carried out by a set of three agents who propose three different sets of plans to support a selected precondition Φ during the dialogue-based search in the space, and where H_1 represents the amount of proposed plans by the first agent, idem for H_2 and H_3. .

[5]The number of agents will be defined at the beginning of the problem.

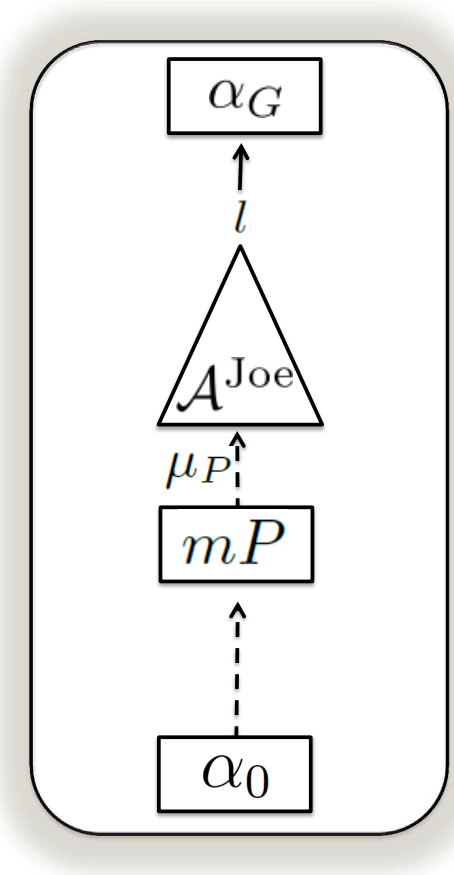

Figure 3.10: A refinement plan $\Pi_r^{(n,i)}$.

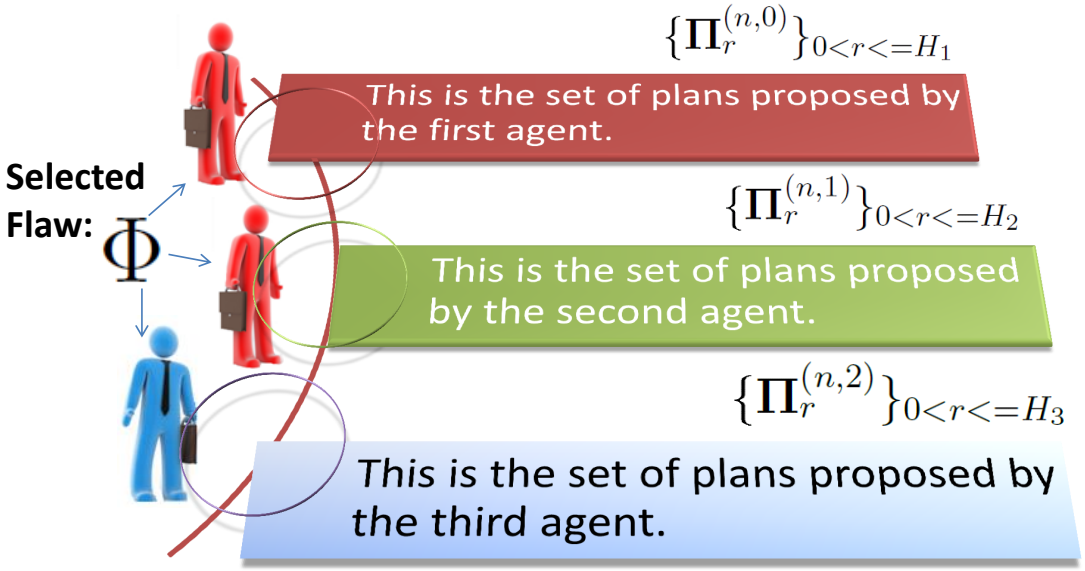

Figure 3.11: Agents proposing different plans.

4

Validation

In this chapter we develop a case study of travel between two cities from Taiwan, using the multi-agent DeLP-POP proposed in this work with the aim to validate our proposal. A scenario analysis, a scenario specification, and finally a scenario implementation of Cooperative Planning, using the different types of dialogue (see section 3) will be detailed in this section.

4.1 Representation of actions with defeasible effects

Suppose we want to represent an action α having irrevocable effects $p_0, p_1, \ldots$ as well as n defeasible effects $d_0, d_1, \ldots$, which are defeated by conditions $d'_0, d'_1, \ldots$ respectively. At its turn, these arguments might be defeated by other arguments (thus restoring the defeasible effect after the action's execution), and so on.

We represent this action α as follows:

- introduce an instrumental irrevocable effect μ' (meaning α *was just executed*),

- then we define $\mathsf{X}(\alpha) = \{p_0, p_1, \ldots, \mu'\}$, and,
- and expand the set of rules Δ with $\{d_k \multimap \mu'\}_{k<n} \cup \{\overline{d_k} \multimap \mu', d'_k\}_{k<n}$.

Following this way it is not necessary to specify all the preconditions of an action. This way multi-agent DeLP-POP may code actions so as to deal with the qualification problem[1].

4.2 Introduction to the scenario

The next example (see Figure 4.1)[2] shows a scenario to Cooperative Planning. There are three different locations in this scenario *Bejing*, *Fuzhou* and *Taipei*. Our multi-agent systems is composed of two agents, *Joe* and *Ann*, who wish to travel to *Taipei* to attend the AAMAS conference as invited speakers. As can be seen, there are several direct or indirect connections between *Bejing* and *Taipei*: via car and ship, train and ship, or plane. The agents, the car, the train and the plane are initially located at *Bejing*, and the goal (G) is to have the two agents at *Taipei* subject to the restriction that they must always travel together. Agents have different knowledge and two pieces of information from each agent can appear to be contradictory. Let's assume that *Joe* uses TV as a source of information, but *Ann* prefers Internet to keep up to date, and both agree in finding a plan that minimizes the time units.

4.3 Scenario specification

For $x \in \mathsf{Ag} = \{Ann, Joe\}$, $\mathbb{M}_x = ((\Psi_x, \Delta_x), A_x, G)$, i.e., two agents are considered[3]:

The following abbreviations can be assumed:

- $l1$, $l2$, $l3$ - *Bejing*, *Fuzhou* and *Taipei*,
- *car*, *tra*, *pl*, *shi* - a car, a train, a plane, a ship,
- r, rl, al, ml - a road, a railway, an airline company, a maritime line,

[1] The qualification problem [Ginsberg and Smith, 1988] is concerned with the impossibility of listing all the preconditions required for a real-world action to have its intended effect

[2] Get Directions on Google maps, http://maps.google.es

[3] We consider propositional STRIPS planning representation, and the default proposition (*have* p) to any literal p that does not have an associated proposition.

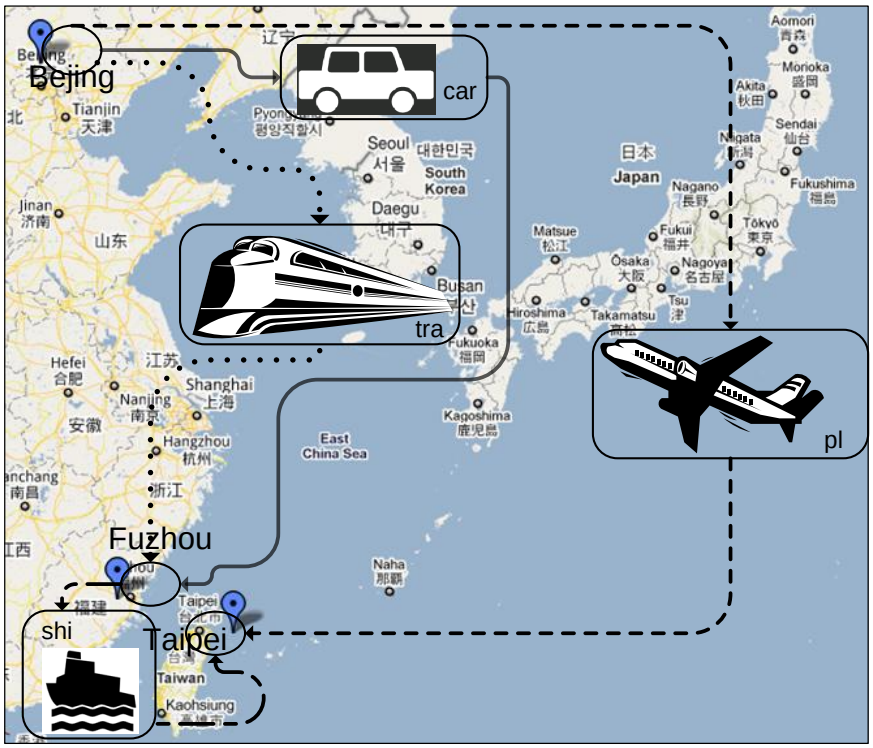

Figure 4.1: Scenario of the application example

- bw, sn, wg, ss - bad weather, snow, wind gusts, stormy sea,
- br, ll, esf, aeo - bad railroad, landslides, electrical supply failure, airplane engines work well,
- va, ds, ip, gw - volcano ash cloud, dangerous situation, risk of increased pollution, contribution to global warming,
- h, tj, $kudTV$, $kudI$ - holidays, traffic jam, kept up to date by TV news, kept up to date by Internet news,
- μ_C, μ_P, μ_T, μ_S - moved car, moved plane, moved train and moved ship

Following, we present the objects defined in this problem:

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

$$A_{Ann} = \left\{ \begin{array}{l} 4.\ \mu_T \xleftarrow{mT} \{(link\ rl\ l1\ l2),\ (at\ tra\ l1), \\ (at\ \mathsf{Ag}\ l1)\} \\ 5.\ \mu_S \xleftarrow{mS} \{(link\ ml\ l2\ l3),\ (at\ shi\ l2), \\ (at\ \mathsf{Ag}\ l2)\} \end{array} \right\}$$

$$G = \left\{\ (at\ \mathsf{Ag}\ l3)\ \right\}$$

1. $mP(pl, j, k)$: moving plane 'pl' from location 'j' to 'k'. It is necessary an airline company to travel from 'j' to 'k', the plane in 'j' and both Joe and Ann in 'j'. Moving a plane takes 2 time unit and 400 cost units.

2. $mT(tra, j, k)$: moving train 'tra' from location 'j' to 'k'. This action takes 6 time units and 200 cost units.

3. $mS(shi, j, k)$: moving ship 'shi' from location 'j' to 'k'. This action takes 3 time units and 100 unit cost.

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

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

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

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

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

In what follows, we explain how our proposal works to obtain a complete plan Π that satisfies the goal G.

4.4 Searching for a solution plan

In what follows, we show, step by step, how the algorithm 1 works. Each number represents the behaviour to a selected plan. As we mentioned in section 3.3.1, arguments will be depicted as triangles. The upper vertex of the triangle will be labeled with the argument's conclusion, and the set of defeasible rules in the argument will be associated with the triangle itself. Sub-arguments will be represented as smaller triangles contained in the triangle which corresponds to the main argument at issue.

1. The planning process starts with an empty plan selected $\Pi^{(0)}$ in step 3 in algorithm 1, essentially $\{\alpha_0 \prec \alpha_G\}$:

    - Since the plan $\Pi^{(0)}$ is only composed by dummy actions, no threats are detected in the dialogue-based plan evaluation (steps from 8 to 11 in algorithm 1) and no no updates are made (steps from 10 to 11 in algorithm 1), and, $\mathsf{MultiThreats} = \emptyset$. Thus, the algorithm 1 jumps to step 15.

- $\mathsf{flaws}(\Pi^{(0)})$ in step 3 returns $(at\ \mathsf{Ag}\ l3)$, and $\Phi = (at\ \mathsf{Ag}\ l3)$ in step 15 in algorithm 1.
- A dialogue-based search in the space of plans starts in step 15 in algorithm 1. *Joe* takes the first turn process and puts forward (among others, not discussed in this example) a refinement plan $\mathbf{\Pi}^{(0,Joe)}$ where:
  $\mathbf{\Pi}^{(0,Joe)} = \{\mathcal{A}^{\text{Joe}}, \alpha_G\}$ and $\mathcal{A}^{\text{Joe}} = (\{(at\ \mathsf{Ag}\ l3)\}, \{(at\ \mathsf{Ag}\ l3) \multimap \mu_P\})$ (see Figure 4.2(a)). *Ann* learns that $\mu_P \in S_{mP}$, and she skip her turn.
- We jump to step 3 in algorithm 1.

2. Suppose $\mathbf{\Pi}^{(0,Joe)}$ is the plan selected, so we move to $(1, \cdot)$ turns.
   - No threats are detected in the dialogue-based plan evaluation, and the algorithm 1 jumps to step 15.
   - $\mathsf{flaws}(\Pi^{(0,Joe)})$ in step 3 returns μ_P, and $\Phi = (\mu_P)$ in step 15 in algorithm 1.
   - A dialogue-based search in the space of plans starts in step 15 in algorithm 1.
     - At $(1, Ann)$ turn[4], *Ann* sends more plans.
     - At $(1, Joe)$ turn, *Joe* refines his previous plan. Since μ_P is not included in $(\psi_{Ann} \cup \psi_{Joe})$, he proposes the action $mP(pl,\ l1,\ l3)$ to support μ_P (see Figure 4.2(b)). Hence, $\mathbf{\Pi}^{(1,Joe)} = \{mP, \mathbf{\Pi}^{(0,Joe)}\}$. Ann learns action mP. We jump to step 3 in algorithm 1.

3. Suppose $\mathbf{\Pi}^{(1,Joe)}$ is the plan selected, so we move to $(2, \cdot)$ turns.
   - A dialogue-based plan evaluation starts. Now its *Ann*'s turn $(2, Ann)$. She finds an argument-argument threat to $\mathcal{A}^{\text{Joe}}$ (i.e. to the support link containing $\mathcal{A}^{Joe}$, and to *Joe*'s plan), based on his initial knowledge of $kudI$. $\mathbf{\Pi}^{(2,Ann)} = \{mP, (\mathcal{B}^{\text{Ann}}, (\mathcal{A}^{\text{Joe}}, \alpha_G))\}$ where $\mathcal{B}^{\text{Ann}} = (\{\sim(at\ \mathsf{Ag}\ l3)\}, \{\sim(at\ \mathsf{Ag}\ l3) \prec \{\mu_P, ds\};\ ds \prec va;\ va \prec kudI\})$ (see Figure 4.2(c)). The initial fact $kudI$ and these rules are learnt by *Joe*. Assume that nothing relevant happens in next $(2, \cdot)$ turns.
   - We jump to step 3 in algorithm 1.

[4] As we mentioned in Section 3.2, (n, i) denotes the turn to the agent i in the iteration n in Algorithm 1.

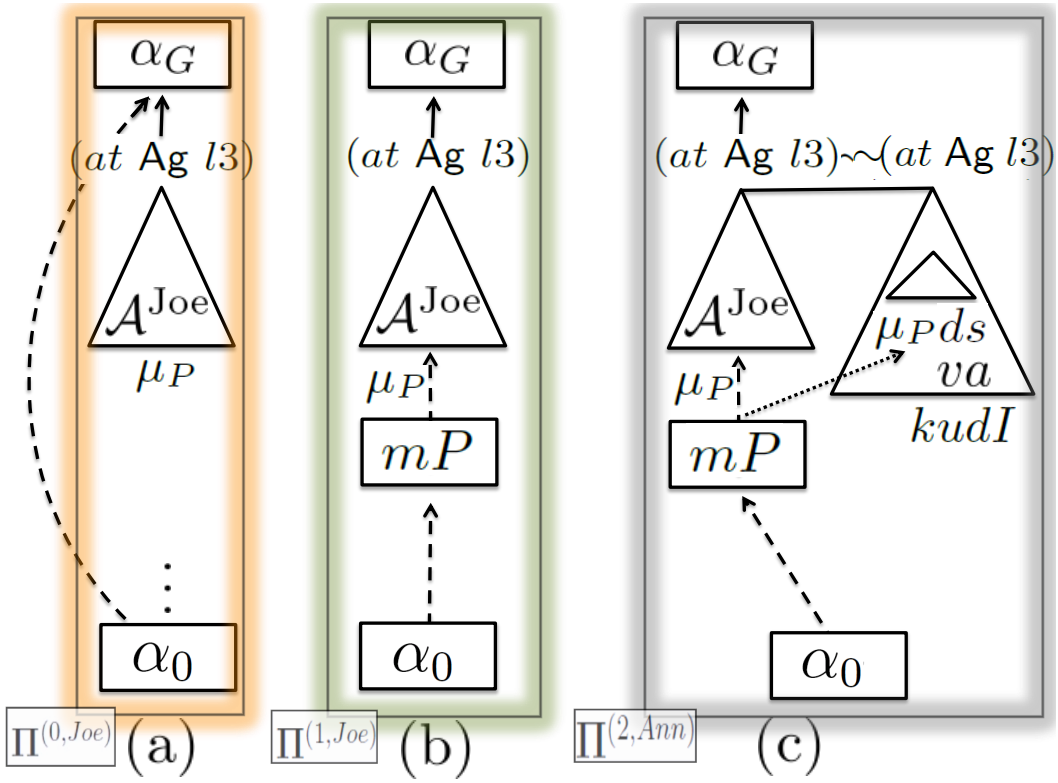

Figure 4.2: Different partial plans for the example scenario (I). (a), (b): Joe's turns and (c): Ann's turn.

4. Suppose that $\mathbf{\Pi}^{(2,Ann)}$ is the plan selected, and we move to $(3,\cdot)$ turns.

   - No new threats are detected in the dialogue-based plan evaluation, and the algorithm 1 jumps to step 15.
   - $\mathsf{flaws}(\Pi^{(2,Ann)})$ in step 3 returns the *argument-argument* threat, and $\Phi = (\textit{argument-argument threat})$ in step 15 in algorithm 1.
   - A dialogue-based search in the space of plans to resolve this threat starts in step 15 in algorithm 1.
     - At Ann's turn $(3, Ann)$, she finds nothing relevant.
     - At Joe's turn (3,Joe), he selects a *Defeat-the-defeater*[5] move against ds, based on her knowledge. $\mathbf{\Pi}^{(3,Joe)} = \{mP, (\mathcal{C}^{\text{Joe}}, \mathcal{B}^{\text{Ann}}, (\mathcal{A}^{\text{Joe}}, \alpha_G))\}$ where $\mathcal{C}^{\text{Joe}} = (\{\sim va\}, \{\sim va \dashrightarrow aeo\})$. It is a

[5]See Section 2.4.1 for resolutions steps to argument-argument threats.

Defeat-the-defeater move since $\overline{\mathsf{concl}(\mathcal{C}^{\text{Joe}})} \in \mathsf{literals}(\mathcal{B}^{\text{Ann}})$) (see Figure 4.3(d)), and $aeo \in S_{\alpha_G}$.

- We jump to step 3 in algorithm 1.

The plan $\mathbf{\Pi}^{(3,Joe)}$ represents a solution plan. However, the algorithm 1 would continue to find all possible solutions.

To sum up, *Joe* suggested to take the plane to arrive to *Taipei*, but *Ann* defeated the proposal because the volcano ashes are expected according to the Internet information, and *Joe* replied that this situation will not affect the flight between *Beijing* and *Taipei*. Then the plan $\mathbf{\Pi}^{(3,Joe)}$ can refined into a solution.

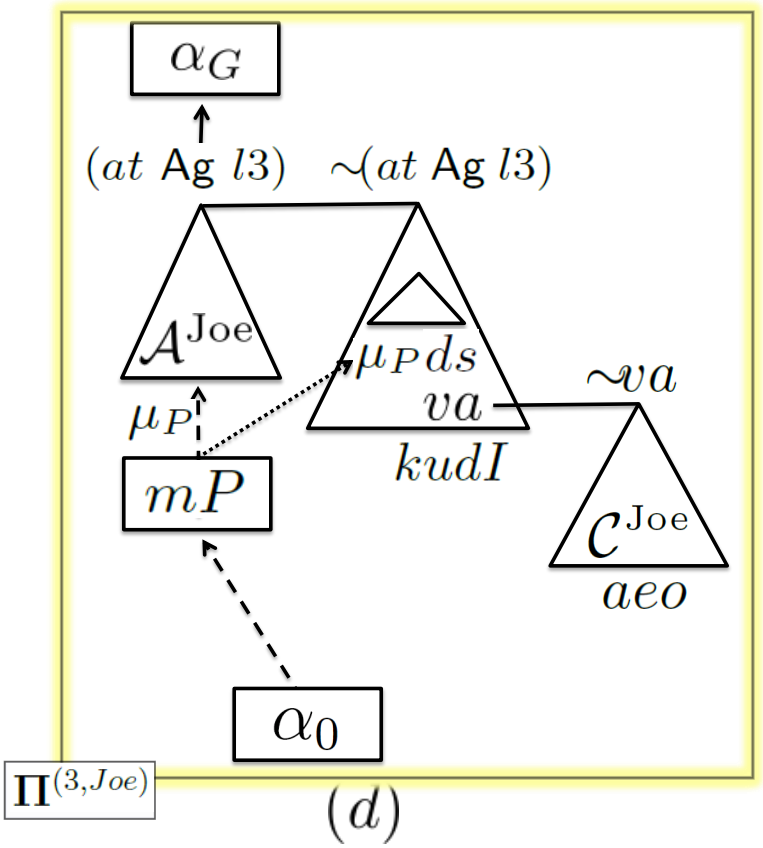

Figure 4.3: Different partial plans for the example scenario (II). (d): Joe's turn.

Figure 4.4 shows the search in the space of partial-order plans for the example scenario, where:

- $\mathbf{\Pi}^{(0,Joe)}$ is the output of the first dialogue-based search in the space of plans,
- $\mathbf{\Pi}^{(1,Joe)}$ is the output of the second dialogue-based search in the space of plans,
- $\mathbf{\Pi}^{(2,Ann)}$ is the output of the first dialogue-based plan evaluation, and,
- $\mathbf{\Pi}^{(3,Joe)}$ is the plan to resolve the argument-argument threat, and is the output of the third dialogue-based search in the space of plans.

Note that, a dialectal tree is implicitly built to each detected argument-argument threat, and in the above case the dialectal tree consists of a single argumentation line $\Lambda = [\mathcal{A}^{\text{Joe}}, \mathcal{B}^{\text{Ann}}, \mathcal{C}^{\text{Joe}}]$.

We do not show the rest of the example, but recall the planning process would not end until the full space of solutions is explored, in order to select the optimal solution (according to either economic or temporal cost).

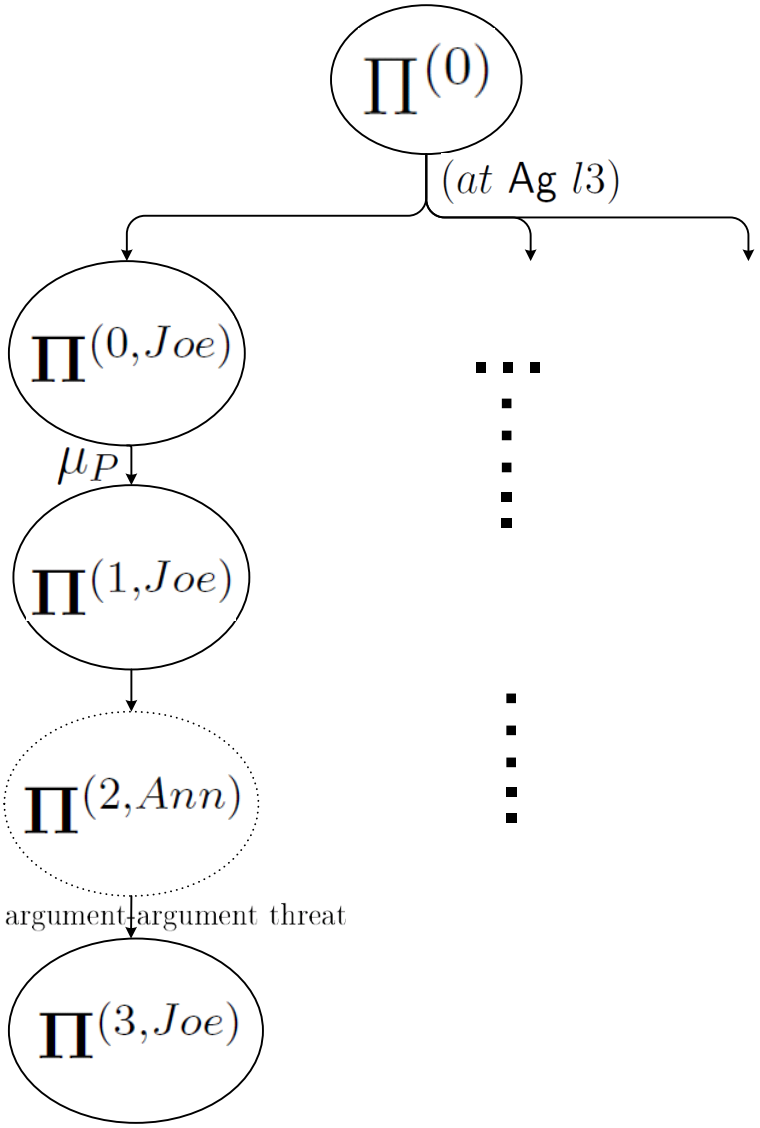

Figure 4.4: Search in the space of partial-order plans for the example scenario.

5

Related Work

In this section, we present a comparison with other works in the literature and justify our choice for multi-agent planning with defeasible argumentation in DeLP-POP.

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

Some systems that build on argumentation apply theoretical reasoning for the generation and evaluation of arguments to build applications that deal with incomplete and contradictory information in dynamic domains. Some proposals in this line focus on planning tasks, or also called practical reasoning, i.e. reasoning about what actions are the best to be executed by an agent in a given situation. Dung's abstract system for argumentation [Dung, 1995] has been used for reasoning about conflicting plans and generate consistent sets of goals [Amgoud, 2003, Hulstijn and van der Torre,]. Further extensions of these works present an explicit separation of the belief arguments and goals arguments and include methods for comparing arguments based on the worth of goals and the cost of resources [Rahwan and Amgoud, 2006]. In any case, none of these works apply to a multi-agent environment. A proposal

for dialogue-based centralized planning is that of [Tang et al., 2010], but no argumentation is made use of. The work in [Belesiotis et al., 2010] presents a dialogue based on an argumentation process to reach agreements on plan proposals. Unlike our focus on an argumentative and stepwise construction of a plan, this latter work is aimed at handling the interdependencies between agents' plans.

On the other hand, we can also find some systems that realize argumentation in multi-agent systems using defeasible reasoning but are not particularly concerned with the task of planning [Thimm and Kern-Isberner, 2008, Thimm, 2009]. All in all, the novelty of our approach is the combination of all these aspects: defeasible reasoning, decentralized planning and multi-agent systems.

6

Conclusions and Future Work

This latter chapter contains the main conclusions of this work, the future lines of research and related publications, produced as a result of this research work.

6.1 Conclusions

In this thesis, we have presented a defeasible argumentation-based model built on the approximation of Garcia et al [García et al., 2008] that extends their work by incorporating multiple agents with partial and contradictory knowledge articulate reasons for and against. Specifically, we presented a decentralized A^* plan search algorithm for multi-agent argumentative planning in the framework of DeLP-POP. This search is implemented as a dialogue between agents, which cooperate to criticize or defend alternative plans by means of defeasible arguments. Only potentially relevant information is exchanged in the dialogue process, which terminates in a provably optimal solution upon which agents cannot disagree.

Along the work, we have introduced the necessary modifications to include a defeasible reasoning into a POP algorithm. This new and enriched model opens up many possibilities to be applied to a multi-agent planning context.

Our work is also related to conformant planning [Hoffmann and Brafman, 2006], an approach to deal with planning with incomplete information in which the purpose is to generate plans given uncertainty about the initial state and action effects, and without any sensing capabilities during plan execution. However, unlike conformant planning, our approach is a powerful planning mechanism for reasoning about contradictory information coming from different sources or agents. In this sense, in the literature of classical planning we can hardly find approaches to deal with contradictory information because, among other reasons, there are very few attempts to extend planning to a multi-agent environment, being a notably exception the work of Brenner and Nebel [Brenner and Nebel, 2009]. Hence, the present work is a novel approach regarding the consideration of incomplete and contradictory information of multiple reasoning entities, i.e. agents.

6.2 Future work

For future work, several directions seem promising: extending the present approach to other multi-agent scenarios, like Argumentation-based Group Recommendation, Argumentation-based Negotiation, or an extension into Temporal Planning.

In this context, we will also study the choice of having non-cooperative agents [Axtell, 2002] in multi-agent DeLP-POP.

Although, multi-agent DeLP-POP guarantees an optimal solution (if some solution exists) by exploring the full space of plans, it would be important to study a heuristic h [Gerevini and Schubert, 1996, Geffner, 2005] $A^{\star}$[1] for selecting of plans, and a different heuristic to select the next flaw, which reduce the search space. This would imply that the number of dialogues between agents would be significantly reduced.

In addition, we are interested in studying a more sophisticated model of learning to the agents.

6.3 Published papers

In this section we present the papers that the author published during his master thesis project.

[1] $A^{\star}$ uses a best-first search and finds the least-cost path from a given initial node to one goal node (out of one or more possible goals).

6.3.1 Papers directly related to this work

This work has been published in an international conferences and a e-book chapter.

1. S. Pajares and E. Onaindia. *DefPlanner: A defeasible argumentation-based planner.* Proceedings of 2nd International Workshop / Special Track on Combinations of Intelligent Methods and Applications (CIMA 2010). pp. 34-42 (2010).

2. S. Pajares and E. Onaindia. *Defeasible Planning through Multi-Agent Argumentation.* Modelling Machine Emotions For Realizing Intelligence, Smart Innovation Systems and Technologies Series, Springer. In Press(2011).

Moreover, a shorter version of this work was submitted to the AAMAS 2011 conference[2], and comments in the rebuttal phase of AAMAS 2011 are quite hopeful.

6.3.2 Other papers

1. I. Garcia, L. Sebastia, S. Pajares and E. Onaindia. *The generalist recommender system GRSK and its extension to groups.* Lecture Notes in Business Information Processing (LNBIP), Springer. In Press(2011).

2. I. Garcia, L. Sebastia, S. Pajares and E. Onaindia. *GRSK: A Generalist Recommender System.* Proceedings of the 6th International Conference on Web Information Systems and Technology. Vol.I. pp.211-218(2010).

[2]http://www.aamas2011.tw/

Bibliography

[Amgoud, 2003] Amgoud, L. (2003). A formal framework for handling conflicting desires. In *ECSQARU*, pages 552–563.

[Atkinson and Bench-Capon, 2007] Atkinson, K. and Bench-Capon, T. (2007). Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171:855–874.

[Atkinson et al., 2006] Atkinson, K., Bench-Capon, T. J. M., and McBurney, P. (2006). Computational representation of practical argument. *Synthese*, 152(2):157–206.

[Axtell, 2002] Axtell, R. (2002). Non-cooperative dynamics of multi-agent teams. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1082–1089. ACM.

[Barrett and Weld, 1994a] Barrett, A. and Weld, D. (1994a). Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112.

[Barrett and Weld, 1994b] Barrett, A. and Weld, D. S. (1994b). Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112.

[Belesiotis et al., 2010] Belesiotis, A., Rovatsos, M., and Rahwan, I. (2010). Agreeing on plans through iterated disputes. In *AAMAS*, pages 765–772.

[Bench-Capon and Dunne, 2007] Bench-Capon, T. and Dunne, P. (2007). Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–641.

[Brenner and Nebel, 2009] Brenner, M. and Nebel, B. (2009). Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems*, 19(3):297–331.

[Carmel and Markovitch, 1996] Carmel, D. and Markovitch, S. (1996). Learning models of intelligent agents. In *Proceedings of the National Conference on Artificial Intelligence*, pages 62–67.

[Cox et al., 2005] Cox, J., Durfee, E., and Bartold, T. (2005). A distributed framework for solving the multiagent plan coordination problem. In *AAMAS'05*, pages 821–827.

[de Weerdt et al., 2005] de Weerdt, M., ter Mors, A., and Witteveen, C. (2005). Multi-agent planning. an introduction to planning and coordination. In *Handouts of the European Agent Systems Summer School (EASSS-05)*, pages 1–32.

[desJardins et al., 1999] desJardins, M., Durfee, E., Ortiz, C., and Wolverton, M. (1999). A survey of research in distributed continual planning. *AI Magazine*, 20(4):13–22.

[Dung, 1995] Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358.

[Durfee, 1999] Durfee, E. H. (1999). *Distributed problem solving and planning*, volume In Gerhard Weiss editor, pages 118–149. The MIT Press, San Francisco, CA.

[Durfee, 2001] Durfee, E. H. (2001). Distributed problem solving and planning. In *Multi-agents Systems and Applications (EASSS 2001)*, volume LNAI 2086, pages 118–149. Springer-Verlag.

[Ferber, 1999] Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 222. Addison-Wesley New York.

[Fikes and Nilsson, 1971] Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

[Ganzha et al., 2005] Ganzha, M., Paprzycki, M., Pirvanescu, A., Badica, C., and Abraham, A. (2005). JADE-based Multi-Agent E-commerce Environment: Initial Implementation. *Analele Universitatii din Timisoara, Seria Matematica-Informatica*.

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

[García et al., 2008] García, D., García, A., and Simari, G. (2008). Defeasible reasoning and partial order planning. In *Proceedings of the 5th international conference on Foundations of information and knowledge systems*, pages 311–328. Springer-Verlag.

[Geffner, 2005] Geffner, H. (2005). Search and Inference in AI Planning. In *Principles and practice of constraint programming–CP 2005: 11th international conference, CP 2005, Sitges, Spain, October 1-5, 2005: proceedings*, page 1. Springer-Verlag New York Inc.

[Gerevini and Schubert, 1996] Gerevini, A. and Schubert, L. (1996). Accelerating partial-order planners: Some techniques for effective search control and pruning. *Arxiv preprint cs/9609101*.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning. Theory and Practice*. Morgan Kaufmann.

[Ginsberg and Smith, 1988] Ginsberg, M. and Smith, D. (1988). Reasoning about action II:: The qualification problem. *Artificial Intelligence*, 35(3):311–342.

[Hoffmann and Brafman, 2006] Hoffmann, J. and Brafman, R. I. (2006). Conformant planning via heuristic forward search: A new approach. *Artif. Intell.*, 170(6-7):507–541.

[Hulstijn and van der Torre,] Hulstijn, J. and van der Torre, L. Combining goal generation and planning in an argumentation framework. In *Proc. Workshop on Argument, Dialogue and Decision at the International Workshop on Non-monotonic Reasoning (NMR'04)*.

[Penberthy and Weld, 1992a] Penberthy, J. and Weld, D. (1992a). UCPOP: A sound, complete, partial order planner for ADL. In *proceedings of the third international conference on knowledge representation and reasoning*, pages 103–114. Citeseer.

[Penberthy and Weld, 1992b] Penberthy, J. and Weld, D. (1992b). UCPOP: A sound, complete, partial order planner for ADL. *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 103–114.

[Prakken et al., 1997] Prakken, H., Gordon, T., Walton, D., Bench-Capon, T., Bex, F., van den Braak, S., van Oostendorp, H., Prakken, H., Verheij, H., and Vreeswijk, G. (1997). Logical tools for modelling legal argument: a study of defeasible reasoning in law. *Dordrecht, Boston*.

[Prakken and Sartor, 1997] Prakken, H. and Sartor, G. (1997). Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1).

[Rahwan and Amgoud, 2006] Rahwan, I. and Amgoud, L. (2006). An argumentation-based approach for practical reasoning. In *International Workshop on Argumentation in Multi-Agent Systems (ArgMAS)*, pages 74–90.

[Ralston et al., 1993] Ralston, A., Reilly, E., Hemmendinger, D., and (Firm), X. (1993). Encyclopedia of computer science.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial intelligence: a modern approach*. Prentice hall.

[Simari and Loui, 1992] Simari, G. and Loui, R. (1992). A mathematical treatment of defeasible reasoning and its implementation. *Artificial intelligence*, 53(2-3):125–157.

[Tang et al., 2010] Tang, Y., Norman, T., and Parsons, S. (2010). A model for integrating dialogue and the execution of joint plans. pages 60–78. Springer.

[Thimm,] Thimm, M. Realizing Argumentation in Multi-Agent Systems using Defeasible Logic Programming.

[Thimm, 2009] Thimm, M. (2009). Realizing argumentation in multi-agent systems using defeasible logic programming. In *International Workshop on Argumentation in Multi-Agent Systems (ArgMAS)*, pages 175–194.

[Thimm and Kern-Isberner, 2008] Thimm, M. and Kern-Isberner, G. (2008). A distributed argumentation framework using defeasible logic programming. In *International Conference on Computational Models of Argument (COMMA)*, pages 381–392.

[Walton, 1996] Walton, D. (1996). *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates, Mahwah, NJ.

[Weld, 1994] Weld, D. (1994). An introduction to least commitment planning. *AI magazine*, 15(4):27.

[Wooldridge, 2009] Wooldridge, M. (2009). *An introduction to multiagent systems*. Wiley.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152.