Universitat de Barcelona (UB)
Facultat de Filosofia
Programa de doctorat en Lògica Pura i Aplicada (2011-2013)
Pure and Applied Logic H0C03
Departament de Lògica, Història i Filosofia de la Ciència

PhD Thesis

Logical planning in Temporal Defeasible and Dynamic Epistemic Logics: the case of t-DeLP and LCC

Pere Pardo Ventura

Advisors:
Lluís Godo Lacasa (IIIA-CSIC)
Mehrnoosh Sadrzadeh (Oxford University)

Tutor:
Ramon Jansana Ferrer (Univ. de Barcelona)

SETEMBRE 2013

A la meva família i amics.

Als meus millors mestres.

Acknowledgments

La tesi que el lector té a les mans no hauria estat possible sense el suport de moltes persones, així com el d'unes quantes institucions.

En primer lloc, voldria agraïr el suport de la meva família, començant pels meus pares Pere i Carme, així com el que vaig rebre dels meus avis Josep i Montserrat, Emili i Montserrat; també agraeixo el suport dels meus germans Emili i Elisa, Mia i Olga (i la Júlia!), i el de tots els cosins, tiets, tietes i oncle.

En segon lloc, vull agraïr també el suport dels meus amics més immediats, Aleix Ojeda, Jaume Clavé, Xavi Miret, Andreu Machuca, Ruben Calvo, Lourdes Bohé, Ricard Roig, Núria Ravents, Milkah Gutiérrez, Carlos Camins, Joan Martí, Núria Galofré, Carles Castillo, Omar Merino i Sara Qadar, amb qui he compartit calçotades i partits de futbol. Gràcies també a la Laura Casademont per la seva hospitalitat, i al Fèlix Llopart i a la Cristina S. pel seu suport.

Pel que fa a la vessant acadèmica, en primer lloc voldria reconèixer i agraïr la paciència, dedicació i bons consells dels meus directors de tesi, el Lluís Godo i la Mehrnoosh Sadrzadeh. No costa gaire d'imaginar què se n'hauria fet d'aquesta tesi sense ells. Gràcies també al meu tutor, en Ramon Jansana, per la seva ajuda i supervisió.

En segon lloc, voldria donar les gràcies també a l'Eva Onaindia i el Sergio Pajares, de la Universitat Politècnica de València, sense els quals bona part d'aquesta tesi tampoc hauria estat possible.

També m'agradaria esmentar totes les persones de l'IIIA que he tingut el plaer de conèixer i de qui he pogut aprendre tantes coses. Molt especialment, Carles Sierra, Ramon López de Mántaras, Pablo Noriega, Francesc Esteva, Marco Cerami, Félix Bou, Pilar Dellunde, Àngel García-Cerdanya *el Sueco*, Enrico Marchioni, Carles Noguera, Tommaso Flaminio, Pedro Messeguer, i la Teresa Alsinet de la UdL. Un agraïment també a les famílies de'n Lluís Godo, Carles Sierra, Ramon López de Mántaras, i Francesc Esteva, i especialment un record per al Marc Esteva i la Noëlle, a qui vaig tenir l'honor de conèixer.

Vull agraïr també els professors del(s) Dept. de Lògica de la UB, de qui tant he après durant els estudis de llicenciatura i de màster. Molt especialment, els professors Ramon Jansana, Ignasi Jané, Joan Bagaria, Calixto Badesa, Enrique Casanovas, Juan Carlos Martínez, Rafel Farré (de la UPC), Ventura Verdú, Josep Macià, Manuel Pérez de Otero, i Manuel García-Carpintero.

Finalment, també he tingut la sort de conèixer molts investigadors i estudi-

ants d'arreu del món durant tots aquests anys. M'agradaria esmentar especialment en Guillermo R. Simari, Ana Casali, David Pearce, Sasha Ossowski, Levan Uridia, Dick Walther, Barteld Kooi, Alexandru Baltag, Wieve van der Hoek, Colin Howson, Nicholas J.J. Smith, Thomas Jech, Hugh Woodin, Petr Hàjek, Chris Fermüller, Ondrej Majer, Raquel Fernández, Petr Cintula, Didier Dubois, Henri Prade, Peter Gärdenfors, Samson Abramsky, Jeremy Seligman, Jeff Paris, Wang Qin, Ilan Frank, Marjon Blondeel, i finalment Bob Coecke, Chris Heunen i James Vicary.

Aquesta llista d'agraïments no estaria completa sense esmentar els companys de despatx amb qui he pogut compartir les pors i esperances típiques dels estudiants de doctorat. De l'IIIA, la Mari Carmen Delgado, Toni Penya-Alba, Xavi Ferrer, Julián Salas, Jesús Giráldez, Marc Pujol, Víctor Bellón, Àngela Fàbregues, Andrew Koster, Amanda Waldermert, Thomas Trescak, Tito Cruz, Isaac Pinyol, Oguz Mulayim, Meritxell Vinyals, Dani Villatoro, Pablo Almajano, Arnau Ramisa, Arturo Ribes, Norman Salazar, Gunnes Baydin, i la resta d'estudiants de l'IIIA amb qui he coincidit en els darrers anys. També el suport de molts investigadors de la IIIA, com ara en Felip Manyà, Sandra Sandri, Marco Schorlemmer, Enric Plaza, Jesús Cerquides, Juan Antonio Rodríguez *JAR*, Josep Puyol, Pere García, Gonçal Escalada i la resta d'investigadors de l'IIIA, així com el de tots els treballadors de l'IIIA i els companys de la UDT.

De la UB, voldria donar les gràcies a en Dan López de Sa, Sònia Roca, Gemma Celestino, Marta Campdelacreu, José Gil, Víctor González, Óscar Cabaco, Joan Trench, Marc Canals, Adán Cassán, Ana González i Jordi Guzmán, així com als companys de màster Andrés Stissman, Neus Castells i Josep Pons.

Finalment, de la UAB, voldria esmentar l'Anna, les dues Raquels, els tres Xavis i la Jèssica, amb qui he compartit tantes converses i avaries als trens.

Aquesta tesi tampoc hauria estat possible sense el suport econòmic dels diferents projectes: AT *Agreement Technologies* (CSD2007-022 Ingenio 2010); LOCOMOTION corresponent al projecte EUROCORES FP006 LoMoReVI (FFI2008-03126-E/FILO); ARINF (TIN2009-14704-C03-03) del MICINN; i el projecte 2009-SGR-1434 de la Generalitat de Catalunya. Agraeixo també el suport de l'Institut d'Investigació en Intel.ligència Artificial (IIIA-CSIC), de la Universitat de Barcelona, i també el suport de la Oxford University durant la meva visita acadèmica l'estiu del 2011.

En cap cas, cap de les persones o institucions esmentades més amunt pot ser considerada responsable dels errors que hagin pogut quedar en aquesta dissertació, però sí en canvi dels encerts que aquesta pogués contenir.

Pere Pardo
Moja - Olèrdola
26 d'Agost de 2013

Abstract

The design of artificial agents -that autonomously decide *what to do* in order to fulfill their goals- has become an important problem at the intersection of applied logic, computer science, game theory and artificial intelligence. This problem is just a rephrasing in contemporary terms of some older questions on *practical reasoning* and *rationality*, with an added emphasis on the need for social abilities in multi-agent systems.

The topics of this thesis are logics and methods for planning. Historically, these two areas independently introduced the same agency-related concepts (actions, time, causality, observations, beliefs, non-determinism, intention), but they developed completely different tools and models for them in order to solve the tasks of inference and plan search (e.g. logical calculi and search methods).

As a consequence, it is difficult to obtain formal theories for practical reasoning that combine the strengths of the two fields for practical reasoning. Thus, for example, the tools used in planning do not easily generalize to planning problems demanding sophisticated reasoning tasks. At its turn, logics have also experienced considerable difficulties to accomplish tasks that plan search algorithms routinely do. Combining the strengths of the two areas would greatly extend the class of scenarios that can be addressed by self-motivated artificial agents.

The aim of this thesis is the study of methods for extending a logic into a planning system (with the expressivity of this logic), and the use of planning techniques to solve practical goals in a logically sound way. To this end, two particular logics are considered: temporal defeasible logic programming and dynamic epistemic logics. These logics are respectively oriented towards the physical or causal aspects of actions, and their epistemic and social dimensions. Although we focus on those particular logics, the proposed techniques seem to generalize to other logics with similar characteristics.

Contents

Preface

The topic of this dissertation is the study of some ways in which logics can be extended into formal models of practical reasoning, using ideas and tools from the area of automated planning. This is shown for two particular logics, called t-DeLP Temporal Defeasible Logic Programming and LCC Logics of Communication and Change. As with many other applied logics in the literature, each of these two logics defines a (meaningful) planning system: first, the propositions of the logical language express world states, planners' goals and also the effects and preconditions of actions; and second, the dynamic transitions between states that model the execution of actions are given by the relation of logical consequence. The class of planning problems in some given logic, i.e. finding a plan that enforces some goals from a given initial state, is thus defined according to this logic. The main contributions of this thesis consist in the study of plan search techniques for the t-DeLP and LCC logics; in each case, these techniques must take into account the particular kind of logical reasoning involved in the construction and verification of solution plans.

A motivation for such an approach to practical reasoning is the possibility of combining the strengths of both planning (e.g. heuristic search) and logics (expressivity and reasoning power). An important part of this thesis is devoted to show how (or to which extent, at least) this can be done for these particular logics, t-DeLP and LCC. Since the approach for practical reasoning followed in this thesis is not unique, it also remains to be seen that it makes sense, in light of other methods for practical reasoning existing in the literature. In this Preface, we try to motivate the present approach also by comparing it to those methods from the two areas of logic and planning.

Logic, or the formal study of valid reasoning, has proved quite successful in solving problems that can be encoded as tasks of propositional reasoning. In applications of logic, these inference tasks are performed by an agent (human or software) trying to decide whether some propositions follow from a knowledge base (a set of non-logical axioms). In most logics, these inference tasks can generally be read as the tasks of an agent trying to decide *what to think* (or *what to believe*).

Since Aristotle, it has been a matter of debate whether logical reasoning includes, as a particular case, practical reasoning. That is, whether the task of deciding *what to do* reduces to that of deciding *what to think*. Practical

reasoning, in its many forms, seems to be ultimately inspired by the following general form of means-ends inference:

if	*the agent desires that B*
and	*the agent knows that (doing) A implies/ causes B*
then	*the agent intends that A.*

Real-world problems, though, exhibit features that suggest that this task is not to be easily accomplished. For example, by applying twice the above form, one obtains the Short-sighted Suitor problem (adapted from [78]):

if I had money she would agree to my proposal for marriage
robbing her is a means to having money
if I robbed her she would agree to my proposal for marriage

This example contains some aspects addressed in either part of the thesis. It can be seen (1) as an illustration of the ramification problem, i.e. the problem of computing the indirect effects of an intention or plan (dealt with in Part I), but (2) it also exhibits epistemic aspects that are essential for a successful plan (studied in Part II); thus, for example, if the suitor proceeds with this plan, it should include some actions like wearing a mask, distort his voice, not to call her by her name, and so on. Finally, this example also illustrates (3) a related area of research, that of multi-agent systems.

Multi-agent systems have become an important paradigm in artificial intelligence [54, 151] and computer science. Some aspects and applications related to the existence of multiple agents are also discussed in this thesis, mainly from the standpoint of cooperative multi-agent planning. In this respect, the logic-based planning systems proposed in each part of the thesis can at least be extended or applied to problems of this kind.

In the above debate on the reducibility of practical reasoning to logic, the existing formal systems in the literature can be broadly classified into two positions:

- *reductionist* position: a task in practical reasoning is an inference task (in some appropriate logic); or a possibly infinite set of inference tasks
- *non-reductionist* position: practical reasoning does not essentially (or in practice) consist in logical reasoning

A strong reductionist position, for example, seems to have been assumed by the proponents of logics of intentions (or motivational attitudes, like desires or goals). These logics try to capture the above form of means-ends reasoning solely by logical means; for example, in the belief-desire-intention BDI logics. An important advantage to the reductionist approach is the already existing literature on logics for different concepts related to agency: actions, time, causality, observations, belief, non-determinism, uncertainty, and so on. See Section 5.4 for a more detailed review of these logics.

At the opposite end, the areas of decision theory, planning, game theory, and the like, did not initially pay much attention to advances in logical reasoning, and proposed their own models for the above agency-related concepts. In classical planning, for example, the task is to find a list of actions that lead to a goal state from the initial state. These states, goals and actions –preconditions, effects– are described only by logical atoms and their negations (and classical implications for conditional effects). Thus, modal logics or non-monotonic logics cannot immediately be used in order to represent or reason with other aspects of states, goals and actions. This is also the case, in practice, for contemporary planners allowing for first-order representations of states and goals. (On the other hand, game theory is much more disconnected from logic, since it fully abstracts from any propositional representation within states, goals and actions. Thus, the algorithm never knows what makes a given state desirable, or why some actions lead to a given particular state.)

The source of disagreement between the two positions can be traced back to the two tasks involved in practical reasoning. Practical reasoning, it has been claimed, decomposes into two tasks: the generation of possible intentions (or plans, or strategies), and the selection of one of these intentions. In this sense, the confronted positions put the emphasis on either the generation or the selection tasks. Thus, logics, on the one hand, can express or generate plans involving logical concepts and use their syntactic or semantic tools to evaluate the actual effects of these plans. Planning algorithms, on the other hand, take advantage of embedding the selection task within the (step-wise) generation of plans, often with specialized selection mechanisms.

The goal of this thesis is to seek a compromise between these two approaches to practical reasoning. Thus, plans are based on some logic, but their construction, made step-by-step, can involve selections that are external to the methods of the logic. This compromise is reached by renouncing to logical models of intentions, and hence the promise of higher-order practical reasoning tacitly made by the language of logics of intentions.[1] From the point of view of planning, the price is in terms of the efficiency of existing planning algorithms that solve problems in the planning systems studied in the literature.

By this, we do not claim that a strong reductionist position in this debate is wrong, but simply current proposals along this line do not seem completely satisfactory. (In comparison, it is relatively straightforward to apply either of the methods proposed here to some suitable logic –without intentions– in order to obtain a planning system.)

Indeed, a major challenge for a logic of intentions is that practical reasoning, in its many forms, seems to be deeply non-monotonic, as noted e.g. by [103]. Thus, if an agent expands its beliefs, its (knowledge of its) own abilities or its goals, the result can be a completely different intention. If this intention is to be generated purely by means of logical inference, then the logic might well

[1] A language with intention modalities permits to express, for example, *I believe that you want me to believe that you do not want my money*; it should also permit agents to intend propositions with nested modalities like the former.

be non-monotonic. The intuition of this issue seems to have provoked different reactions in the literature.

First, some logics clearly oriented to practical reasoning simply restrict themselves to the monotonic fragment of practical reasoning. This fragment includes the generation of possible intentions, and excludes the selection step. This is the case, for example, of graded BDI logic [37], that infers possible intentions with associated degrees of desirability, leaving to the end-user the task of selecting some optimal intention. (Similar comments apply to logics of strategic ability, with logical theorems be of the form *the coalition has some strategy that forces this proposition*. The utility of the existing strategy depends on that of the proposition, both to be externally judged.)

A second reaction has been to embrace the non-reductionist approach and introduce elements of planning as part of the implementation of these logics. This is the case of some BDI logics, where the reasoning agent is assumed to be endowed with a module for plan generation, or simply with a library of plans. See [151], [37]; see also [28] for a dynamic epistemic logic viewpoint on this issue. The present approach also falls in this class.

Finally, a third reaction has been to embrace non-monotonicity for the selection of intentions, see [125] for an argumentation-based approach, and [103] for a defeasible extension of BDI logics, using the defeasible logic in [102], [10]. (Note that, even if Part I makes use of argumentation tools and defeasible logic, these are only used to generate plans, not to select among them. The latter task is not to be carried out by the logic but by the planning algorithm.)

The present dissertation can rather be seen along the line of the second kind of reaction. A major difference, though, is that the selection task is carried out by the planning algorithm, so no intention modalities need to be assumed in the logical language. Instead, the logics selected in this dissertation focus on other aspects of reasoning about agency. The two logics t-DeLP and LCC studied in each part of this thesis are motivated by the following considerations.

On the one hand, most planning systems assume classical logic, i.e. monotonic logics, as their base logic. Different issues like the frame problem have long been identified that prevent sophisticated reasoning about actions in this kind of logics and planning systems. For scenarios oriented to temporal causal reasoning, in Part I we propose a temporal extension of defeasible logic programming DeLP[61]. The resulting temporal defeasible logic programming system t-DeLP allows to encode temporal processes as arguments (proofs) and, in case of a conflict due to their interaction, select among the former according to some structural properties of the arguments representing them.

On the other hand, different (monotonic) logics have been studied in order to reason about agent-related notions. Besides initial attempts based on classical first-order logic, the advantages of modal logic over first-order logic in representing agent-related notions have lately been recognized and are generally accepted [92, 90, 141]. Along this line, in Part II we focus on dynamic epistemic logic. These study the epistemic aspects of multi-agent systems, including the epistemic effects (and preconditions) of actions. To this end, we selected an

expressive dynamic epistemic logic, the family of LCC logics of communication and change [139].

We hope the present studies can be seen as particular contributions to the two areas of logic and planning. From the point of view of logic, the main contributions are proofs of standard logical properties (logical soundness, completeness) or standard argumentation-theoretical properties (consistency, closure). From the point of view of planning, the main contribution is an increase in the expressivity of (the languages) of existing planning systems, or in the reasoning power of the logics underlying these planning systems (see below). Part I is mostly an example of the latter w.r.t. temporal planning, while Part II is an example of both, w.r.t. classical planning or planning with partial observability. Finally, some issues in multi-agent planning are also considered.

Overview and structure of the dissertation

The chapters in each part of the thesis roughly follow the same structure: first, a logic is introduced; then, this logic is seen to induce (by introducing actions) or contain some kind of state transition system, where the effects of actions and plans can be computed. The addition of goals permits to define a planning system based on this logic. The planning system that will result mainly depends on two aspects: the nature of actions and the direction of plan search (whether we opt for a forward or a backward approach to build plans). Then some planning algorithms –mostly based on Breadth First Search– are proposed for the corresponding planning systems. Finally, these algorithms are shown to be sound and complete w.r.t. the space of plans. In other words, if given some planning problem, the algorithm terminates with a plan, then this plan is a solution for that planning problem (soundness). And conversely, if a solution exists, the planning algorithm terminates with such a solution plan (completeness). A more detailed summary of the different chapters, and their relations to the list of publications is given next.

Chapter 1 In this first chapter we present a general framework for the t-DeLP temporal defeasible logic programming, as well as a detailed study of its argumentation-theoretic properties for two classes of logical programs, called simple and mutex. As in logic-based argumentation, the idea is to replace the notion of proof (from monotonic logics) by that of argument, and then compare the existing arguments for or against a claim, to decide about the truth-value of this claim (true, false, undecided). Thus, the t-DeLP logic programming system focuses on non-monotonic temporal reasoning for answering simple queries (temporal literals) on future states, according to some logical program or knowledge base. The t-DeLP system was first presented in [106] for simple programs, and an expanded and revised version appeared later as [108], which also dealt with mutex programs. This Chapter is mostly based on [108], except for the general definition of defeat, and a rephrasing of the section on mutex programs in a more logical vein. A related contribution, not included in this dissertation, is [70], on

a possibilitic extension of t-DeLP.

Chapter 2 In this chapter, a multi-agent planning system built on t-DeLP is proposed. First, temporal actions are externally introduced, as in planning. A state transition system is presented, which combines the traditional action update with t-DeLP reasoning. A planning system based on t-DeLP is then defined for planning domains of the form *(t-DeLP program, actions, goals)*. This Chapter focuses on centralized planning, where a planner algorithm generates a joint plan for the set of executing agents, that enforces the temporal goals. The main contribution of this paper is the proof that Breadth First Search (BFS) is sound and complete for (centralized) plan search, both for forward and backward search. This Chapter is entirely based on the paper [107].

Chapter 3 We study in this chapter a decentralized version of the algorithms from the previous Chapter 2. The central planner from Chapter 2 is replaced by a set of planners, who agree upon the set of goals, and aim to agree as well upon a joint plan for these goals. Each planner, though, is assumed to have its own knowledge base and abilities. This Chapter studies a dialogue-based plan search algorithm for this task, where agents exchange new suggestions or evaluations for plans The main contributions in this Chapter are the preservation of soundness and completeness of BFS centralized planning to the present dialogue-based algorithm. This Chapter is essentially based on [109], though many of the ideas were first studied in the DeLP-POP framework [110] and [111].

Chapter 4 In this chapter, we review the dynamic epistemic logics used in later chapters for the purpose of planning. These logics, proposed in [139] are called Logics of Communication and Change (LCC), a family of DEL logics which capture and generalize most of the standard DEL logics in the literature. In general, dynamic epistemic logics are modal logics with both dynamic modalities for actions, and epistemic modalities for agents' knowledge. This Chapter is mostly based on

[139] J. van Benthem, J. van Eijck and B. Kooi. Logics of communication and change, *Information and Computation*, 204: 1620–1662 (2006)

Chapter 5 In this chapter we study a planning system for backward search in an arbitrary LCC logic. This planning system is simply introduced as usual by a tuple *(initial state, actions, goals)*, where now the initial state and goals are formulas of the LCC logic, and the available actions are a subset of the action model. Since the assignments considered in LCC logics represent deterministic actions, any such LCC logic induces a deterministic planning system. In the present Chapter, then, we focus on Breadth First Search for backward deterministic planning. The main contributions are the proofs for the soundness and completeness of this search method. A first version of this chapter can be found in [113], which makes use of generalized frame axioms for the persistence (or

change) of goals. This technique was replaced in [112] and [114] by syntactic tools from [139]. This chapter is mostly based on the paper [114].

Chapter 6 In this chapter, we propose an extension of LCC logics with non-deterministic actions. This is done by combining the (atomic) deterministic actions of LCC logics with the PDL program constructors of choice and (bounded) composition, denoted $\mathrm{LCC}_{\cup\otimes n}$. These logics extend the language of LCC with modalities for complex dynamic epistemic programs (or plans), which are conditional upon the actual results of their non-deterministic actions. The construction of the $\mathrm{LCC}_{\cup\otimes n}$ logics takes place in a step-wise fashion. We introduce the different elements (composition and non-deterministic choice) in an increasing way. The main contribution of this Chapter is the soundness and completeness of the logics $\mathrm{LCC}_{\cup\otimes n}$, which naturally extend the semantics and axioms of LCC. This Chapter is mainly based upon the paper [114].

Chapter 7 A planning system for an arbitrary $\mathrm{LCC}_{\cup\otimes n}$ logic is introduced. A planning domain is as in Chapter 5, except that certain combinations of (possibly unavailable) deterministic actions logic describe non-deterministic actions available to an agent. For example, tossing a coin decomposes into tossing heads and tossing tails. A non-deterministic planning method for arbitrary $\mathrm{LCC}_{\cup\otimes n}$ logics is proposed, which reduces non-deterministic planning in some $\mathrm{LCC}_{\cup\otimes n}$ logic into a series of deterministic plan searches in the LCC fragment of this logic. The main contribution of Chapter 7 is the proof that the new BFS-inspired algorithm is a sound and complete algorithm for strong non-deterministic planning. That is, a solution plan necessarily leads to some goal state, in any particular execution (or instantiation) of this plan in the initial state. This Chapter is also mostly based on the paper [114].

List of publications related to this dissertation.

Most of the contributions of the present thesis are based on the following publications.

[106] Pere Pardo and Lluís Godo
t-DeLP: a temporal extension of the defeasible logic programming argumentative framework
Proc. of Scalable Uncertainty Management SUM 2011
Benferhat and Grant (eds.) LNAI vol. 6929 pp. 489–503, Dayton, USA (2011)

In this paper, a proposal to extend the language of defeasible logic programming DeLP with explicit time was first considered. The language contains two types of rules: strict and defeasible. Certain aspects of temporal reasoning are taken into account (persistence, future-oriented causality). The contributions of this paper concern the class of simple programs. Basic properties of consistency and closure are proved for this class of t-DeLP programs.

[108] Pere Pardo and Lluís Godo.
t-DeLP: an argumentation-based Temporal Defeasible Logic Programming framework
Annals of Mathematics and Artificial Intelligence, Elsevier (In Press, 2013)

This is an expanded and revised version of [106]. The main difference is the introduction and study of mutex programs. A mutex program is just a simple program extended with strict rules modeling mutex constraints (used in planning systems to capture strong incompatibilities). This class is studied by considering a strengthening of the relation of attack beyond the two-valued case. The class of t-DeLP mutex programs is shown to satisfy the basic properties of consistency and closure.

[70] Lluís Godo, Enrico Marchioni and Pere Pardo
Extending a temporal defeasible argumentation framework with possibilistic weights
Proceedings of the 13th European Conference on Logics in Artificial Intelligence JELIA 2012, Fariñas del Cerro, Herzig and Mengin (eds.) Toulouse, France (2012)

In this contribution, we study a system related to t-DeLP, that extends the language of temporal literals with possibilistic weights. This permits to reason with qualitative uncertainty in the style of PDeLP, a possibilistic version of DeLP. The proposed system, called pt-DeLP results from combining the defeat relations of t-DeLP and PDeLP based on temporal criteria and the strength of beliefs. Two lexicographic orderings on these notions of defeat are studied. The paper shows that under any of these two orderings, pt-DeLP is a conservative extension of t-DeLP, but not of PDeLP.

[107] Pere Pardo and Lluís Godo
An argumentation-based multi-agent temporal planning system built on t-DeLP
Proceedings of the Spanish Conference on Artificial Intelligence CAEPIA 2013 Bielza, Salmerón, Alonso-Betanzos et al. (eds.) LNAI vol. 8109, Madrid, Spain (In press)

In this paper, a multi-agent planning system based on t-DeLP logic programming is considered. The focus is on centralized planning, where a unique planner generates a sound plan (or solution). This paper describes the state transition system that corresponds to extending t-DeLP with temporal actions in the style of planning. The main contribution concerns the soundness and completeness of Breadth First Search in forward and backward planning, though most of the paper is devoted the less trivial case of backward plan search.

[109] Pere Pardo and Lluís Godo
A temporal argumentation approach to cooperative planning using dialogues
Proceedings of the 14th Workshop on Computational Logic in Multi-Agent Systems CLIMA 2013, Leite, Son, Torroni, van der Torre and Woltran (eds.) LNAI vol 8143, pp. 307–324, La Coruña, Spain (In press)

In this contribution, we present dialogues for decentralized planning in t-DeLP. The task of a central planner assumed in [107] is split into a set of autonomous planner-executor agents which share the same set of goals, but otherwise can have different beliefs or abilities. We propose a distributed algorithm for these agents, which basically instantiates a dialogue for the construction of a joint plan, which is seen as a solution from the point of view of each of these agents. These dialogues consist in an exchange of plan proposals and their evaluations, and can be seen as instantiating a Breadth First Search algorithm of [107]. The main contributions is again a proof of the soundness and completeness of this dialogue-based algorithm, which is done by comparing it to a central planner endowed with all the information

[110] Pere Pardo, Sergio Pajares, Eva Onaindia, Lluís Godo and Pilar Dellunde
Multi-agent argumentation for cooperative planning in DeLP-POP
Proceedings of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems AAMAS 2011, Tumer, Yolum, Sonenberg and Stone (eds.) pp. 971–978 IFAAMAS, Taipei, Taiwan (2011)

This contribution is part of a series of papers on the study of dialogue-based algorithms for DeLP-POP [62]. This is a flexible planning system, based on partial order planning for Defeasible Logic Programming [61]. In this paper, dialogues for cooperative problems in DeLP-POP are proposed and shown to preserve the properties of the centralized planning methods from [62].

[111] Pere Pardo, Sergio Pajares, Eva Onaindia, Lluís Godo and Pilar Dellunde
Cooperative Dialogues for Defeasible Argumentation-based Planning
Proceedings of the Argumentation in Multi-Agent Systems workshop ArgMAS 2011, McBurney et al. (eds.) LNCS 7543, pp. 185–204
Taipei, Taiwan (2011)

This paper is a revised version of [110], with several technical changes upon the definition of planning actions. These result in a greatly simplified DeLP-POP planning system. The dialogue-based planning algorithms for this new planning system, and the main results are similar to those of [110].

[113] Pere Pardo and Mehrnoosh Sadrzadeh
Planning in the Logics of Communication and Change
Proceedings of Autonomous Agents and Multiagent Systems AAMAS 2012
Conitzer, Winikoff, Padgham, and van der Hoek (eds.), pp 1231–1232, València, Spain (2012)

This paper contains a first study of planning algorithms, based on Breadth First Search, for a backward planning system built on the Logics of Communication and Change LCC [139]. This algorithm is shown to be sound and complete for the base epistemic language E·PDL without common knowledge. The proof techniques for this results consists in the generation of generalized frame axioms that compute (in a backward fashion) issues of persistence or change for open goals.

[112] Pere Pardo and Mehrnoosh Sadrzadeh
Backward Planning in the Logics of Communication and Change
Proc. of 1st International Conference on Agreement Technologies AT 2012
Ossowski, Toni and Vouros (eds.), pp. 231-245, Dubrovnik, Croatia (2012)

In this contribution, the previous results are extended to the class of planning domains whose goals and knowledge base are formulas in the full language of E·PDL. A simple form of non-deterministic planning is also considered, based on an extension of the dynamic modalities in the language, with composition and choice. Chapter 5 is based on this work.

[114] Pere Pardo and Mehrnoosh Sadrzadeh
Strong Planning in the Logics of Communication and Change
Post-Proceedings of Declarative Agents, Languages and Technologies DALT 2012 Baldoni, Dennis, Mascardi and Vasconcelos (eds.), pp. 37–56, València, Spain (2012)

This paper describes the logics, planning systems and plan search algorithms considered in Chapters 5, 6 and 7 of this thesis. Its main novelty is a more expressive planning system for LCC with non-deterministic actions. The main

contributions are again the soundness and completeness of plan search algorithms for both deterministic and strong non-deterministic planning. The latter non-deterministic planning is shown to reduce to a series of problems in deterministic planning, which can be solved with Breadth First Search. A longer version of this paper is under preparation.

Terminology and notation.

There are several terms and formal symbols that have different meanings among related areas where they occur. For example, the term *program* has different meanings in logic programming (a knowledge base $\mathbb{P}$), in dynamic or epistemic logics (a modality π).

$p, q, r, \ldots$	atomic variables	$\mathsf{e}, \mathsf{f}, \ldots$	actions
Var	set of atomic var.	action	an action
t	transl. function	U	an action model
t	temp. parameter	$[\mathsf{U}, \mathsf{e}]$	action modality
$\langle \ell, t \rangle$	temp. literal	$\varphi, \psi, \ldots$	a formula
$\sim p$	strong negation	$\neg p$	classical negation
$\rightarrow$	classical implication	$\mathsf{pre}(\mathsf{e})$	preconditions of e
$\leftarrow$	strict arrow	$\mathsf{post}(\mathsf{e})$	effects of e
$\prec$	defeasible arrow	A	set of avail. actions
Ag	the set of agents	A'	a set of actions
$\mathbb{P}$	a t-DeLP program	$\mathbb{M}$	planning domain
$\mathbf{M}$	a mutex family	M	epistemic model
π	a plan	π	a PDL modality
$\pi_\varnothing$	the empty plan		
$\pi; \pi'$	composition in PDL	$\pi \cup \pi'$	choice in PDL
$\mathsf{e} \otimes \mathsf{f}$	comp. of actions	$\mathsf{e} \cup \mathsf{f}$	choice for actions
$\mathbb{P}$	a t-DeLP program	$X \cup Y$	set-theoretic union
$\mathbb{P} \diamond \mathsf{e}$	update of $\mathbb{P}$ by e	$M \circ \mathsf{U}$	product update
$\mathbb{P} \oplus \mathsf{e}$	expansion of $\mathbb{P}$ by e	(w, e)	update of w by e
$[\![\varphi]\!]^M$	interpr. of φ in M	$\mathcal{T}_{\mathbb{P}}(\cdot) \sqsubseteq \mathcal{T}_{\mathbb{P}'}(\cdot)$	extension of trees
$\pi^{\mathbb{M}}$	interpr. of plan π in $\mathbb{M}$	$\mathbb{M} \sqsubseteq \mathbb{M}'$	ext. of plan. dom.

$\pi(\mathcal{A})$ or $\pi(\mathsf{e})$	refinement of plan π with argument $\mathcal{A}$ or action e
$\pi_\varnothing(\mathbf{x}_1, \ldots, \mathbf{x}_n)$	a plan as a sequence of refinements
$(x, y, \ldots)$	a tuple, e.g. in $\mathbb{M} = (\textit{init. state, actions, goals})$
$(x, \ldots, x') ^\frown (y, \ldots, y')$	concatenation of sequences $(x, \ldots, x', y, \ldots, y')$
$[\mathcal{A}, \mathcal{B}, \ldots]$	a tuple of arguments or an argumentation line

Part I

Planning in t-DeLP Temporal Defeasible Logic Programming

Introduction

In this first part of the thesis, we introduce and study in Chapter 1 a temporal defeasible logic programming framework, called t-DeLP. This t-DeLP framework is later used to define different t-DeLP planning systems in Chapters 2 and 3.

The t-DeLP system from Chapter 1 is an argumention-based non-monotonic temporal logic programming framework. Its expressivity is restricted to the Horn-fragment of a rule-based language built on a set of temporal literals. Temporal literals are represented as pairs of the form $\langle literal, time \rangle$, expressing that some type of fact *literal* is true at some *time*.

The main novelty of t-DeLP is the combination of defeasible and temporal reasoning of this kind. Defeasible logics consider two types of expressions strict and defeasible. Strict information behaves monotonically, while defeasible information draws presumable (but possibly unsound) conclusions. These conclusions can be challenged by other constructible arguments. An argumentation-based procedure decides which conclusions are to be held.

A logical program in t-DeLP can naturally be seen as the knowledge base of an agent at an initial state (e.g. the current state). A t-DeLP reasoning agent can thus answer whether a fact will occur at a given (future) time, using the available knowledge in her knowledge base.

This t-DeLP logic programming framework is used in Chapters 2 and 3 as the logical foundation for a multi-agent planning system. Indeed, a natural extension of t-DeLP towards (temporal) practical reasoning simply consists in adding a set of temporal actions, and a set of agents which can execute these actions. This extension is considered in Chapter 2. The interactions between t-DeLP reasoning and action update define a t-DeLP notion of state transition systems, where the defeasible effects of executing actions can be computed. Moreover, if we add a set of goals to a t-DeLP state transition system, the result is a t-DeLP (centralized) planning problem. That is, a central planner must find a joint plan to be executed by the executing agents. The central planner will try to build a plan for these goals from her knowledge base (a t-DeLP logic program) and the abilities of the agents. The idea is that the execution of this joint plan by the agents will lead to a state where the temporal goals are satisfied. In Chapter 2, we will focus on the Breadth First Search algorithm (BFS), although other search algorithms might be used instead. In particular, we show that BFS is sound and complete, for both forward and backward t-DeLP planning systems.

Finally, Chapter 3 studies a planning system for t-DeLP that does not assume the existence of a central planner. In this case, each agent is endowed with its own knowledge and abilities. We study in particular the sub-class of problems in cooperative planning, where the agents share a set of goals, and they wish to reach an agreement upon a joint plan, whose execution would benefit them all. This class of problems can be solved with the help of dialogue-based planning algorithms. In particular, the agents start a dialogue to find a correct plan, by proposing plans and discussing them. These algorithms are also shown to be sound and complete, so after this dialogue, the agents either agree upon a joint plan or they acknowledge that no solution exists for their goals.

Chapter 1

t-DeLP Temporal Defeasible Logic Programming

1.1 Introduction

In this chapter we present a temporal defeasible logic programming system, called t-DeLP. This system of logic programming is based on logical argumentation tools, along the line of DeLP by García and Simari [61], but with a specific focus on temporal causal reasoning. As with other logic programming systems in the family of DeLP, the present chapter combines tools from abstract argumentation [52] and its extension with logic-based arguments [121] with a formal relation of preference, specially designed for temporal reasoning. Also as in most logic programming systems, we define (an algorithm for) a notion of logical consequence in, roughly, the Horn fragment of a propositional language. The algorithm takes as inputs a logical program and a query, and returns a yes/no answer to the query.

The main motivation for a temporal extension of DeLP is to define a non-monotonic method to solve queries of the form: *when will a process cease (or start)?* A defeasible argumentation approach to temporal reasoning seems specifically useful when the "discussion" is precisely between the different conjectures about the time a proposition (representing some aspect of the process) will cease to be true. Questions of this type, in particular, require us to modify DeLP in order to address several existing genuinely temporal aspects of reasoning, like persistence. Finally, these goals are to be pursued while keeping the conceptual simplicity of defeasible argumentation intact.

Defeasible reasoning (and, more generally, non-monotonic reasoning) tries to capture common-sense inferences, based on normal expectations. This provides a simple form of reasoning when exceptions are not known to apply; at its turn, the

possibility that exceptions apply is accommodated by the defeasible character of inference. The problem with many of the initial proposals in non-monotonic reasoning is that they are based on difficult and unnatural concepts, while on the other hand, they demand the end-user to define a complex preference relation in addition to the usual task of representing a knowledge base with a set of expressions in the language of the logic.

In argumentation-based logics, the arguments can roughly be seen as minimal consistent derivations. While argumentation based on classical logic programming has a unique symbol $\leftarrow$ for Horn clauses [91], defeasible argumentation considers a rule-based language with two types of rules: strict $\leftarrow$ and defeasible $\prec$. The idea is that if a derivation makes use of defeasible information at some point, then from this point onwards all the intermediate consequences are defeasible; in contrast, when the derivation entirely consists of strict information, then the defeasible consequence relation behaves as in the classical case.

Temporal reasoning has also been studied in the area of temporal logic programming [20], [1]. These works include expressive extensions of classical logic programming, e.g. with temporal operators *next*, *always-in-future*, and so on. Despite the expressivity of these languages, their notion of logical consequence is monotonic, and hence these models for reasoning with persistence or with time in general are somewhat simplistic. In practice, this forces an end-user to represent domain-specific rules (the clauses in the logical program modeling a knowledge base) either in a casual non-modular way, or using cumbersome representations for highly-detailed knowledge bases. The former is problematic since further introductions of more precise rules demand to rewrite many parts of the logical program. The latter demands a huge number of domain-specific rules accounting for each single possible scenario.

The advantages of the present approach based on defeasible argumentation are precisely along the lines mentioned above. On the one hand, argumentation-theoretic tools allow for some conceptual simplicity in the design of non-monotonic logics, as these tools and concepts are naturally inspired by processes of deliberation or discussion among humans. Another advantage is that t-DeLP dispenses end-users with the further task of encoding a notion of preference (among possible inferences), so only the usual task of encoding facts and domain-specific rules is required, as in classical logic programming. This latter advantage is made possible by the definition of a general-purpose relation of preference among arguments. This notion of preference aims to capture temporal causal reasoning among common-sense users or agents. Let us remark that defining a non-monotonic logic with such a unique, formal notion of preference brings this logic closer to the traditional view of mathematical logic; that is, the view that logic is the study of the formal aspects of sound reasoning. In the present case, the purely formal notion of preference is based, as usual, on the logical syntax, but also on some structural aspects of arguments.

In summary, the t-DeLP framework is as follows. The logical language is defined by a set of temporal literals -representing facts occurring at some time-, and (strict or defeasible) durative rules. A temporal logic program consists of

temporal facts and rules, which combine into arguments for further derivable facts. The main motivation for t-DeLP is to reason about interacting processes (modeled as arguments), and then decide which arguments (conclusions) are to prevail. An argument expresses some delay between each premise (cause) and the conclusion (causal effect), thus suggesting how a process might evolve. Since different arguments (process descriptions) might conflict, a dialectical procedure is proposed that decides which arguments prevail. This procedure classifies the arguments (constructible in a logical program) into 'defeated' or 'undefeated'. Finally, the t-DeLP notion of warrant, or defeasible logical consequence, in a given logical program, is defined as the set of conclusions of undefeated arguments that are constructible in this program. The literals which are conclusions of some such undefeated argument are called warranted in the t-DeLP logical program. Warranted literals can be read as a partial description of the future temporal states that will turn up, according to the available information contained in the logical program. A t-DeLP theorem prover can thus be seen as a decision method for queries of the form: does p holds at time t, with possible answers: yes, no, undecided.

Here we do not address, though, the technical details for an implementation of a logic programming algorithm for t-DeLP. The reader is referred to [20], where a resolution-based proof method is studied for temporal logic programming [1]. Since temporal literals can be expressed with the operator *next*, the proposed algorithms for refutation proofs can be used in t-DeLP to build arguments. The remaining argumentation-theoretic machinery to evaluate these arguments in a logical program (that is, the definitions of attack, defeat, argumentation line, dialectical tree, warrant) can simply be built on top of these unification algorithms.

After this brief review, let us expand on the motivations for a defeasible argumentative approach to temporal reasoning. Among the reasons that can be given for the adoption of a defeasible (more generally, non-monotonic) approach is the descriptive parsimony it allows for knowledge bases. This parsimony is in accordance with everyday causal reasoning, where it is standard practice to list only those causes that are uncommon or just specific to the process: e.g. a *spark* caused a *fire*. Causes that usually hold, like *oxygen*, are not mentioned in the explanation (or rules) unless they are false and this explains the non-occurrence of the effect: the *spark* did not start a *fire* because *no-oxygen*. With the help of temporal information, the queries that can be asked are of the form *will p occur at time t?* or more generally about the duration of temporal processes *when will p start/cease to be true?*.

A well-known contribution among argumentation-based defeasible logics is that of García and Simari's [61]. The authors present DeLP, a logic programming formalism based on defeasible argumentation. The question of how to define the defeat or preference relation between arguments is also discussed at length in this work. Inspired by Poole [120], the authors of [134] focus on a formal criteria called *generalized specificity*, which gives preference to arguments with

more premises or more direct rules.[1] But the latter seems at odds with causal reasoning in a temporal setting: we would rather prefer *less direct* rules, i.e. more detailed temporal inferences. [2] Thus, we adapt this and other aspects of DeLP to the temporal case in order to intuitively meet basic intuitions about causal explanations. Some of these differences arise from the temporal asymmetry (past vs. future) of causation: persistence, the attack relation and defeat deserve special attention for the temporal case. As a consequence, the notion of warrant for (temporal) literals is slightly different from that of DeLP as presented in [61].

Structure of the chapter

The chapter is structured as follows. Some preliminaries on notation and knowledge representation are presented in Section 1.2. Then, we present in Section 1.3 a general framework for t-DeLP logic programming. In Section 1.4, we focus first on the study of t-DeLP for simple programs (programs without strict rules), and show that under this restriction t-DeLP satisfies the basic argumentation-theoretic properties, called rationality postulates. Then, in Section 1.5, we extend these results to t-DeLP mutex programs, simple programs which admit strict rules if induced by mutex constraints. The chapter ends with Section 1.6, where t-DeLP is compared with Dung semantics [52]; with the defeat criteria from DeLP (other elements are compared throughout the chapter); and finally with the closely related logic programming framework TDR [14]. After the Conclusions and Related Work sections, an Appendix section contains the proofs of auxiliary results presented in this chapter. The remaining results and proofs, containing illustrative tools in defeasible argumentative logic programming, are presented throughout the text.

1.2 Representing Temporal Change in t-DeLP

Concerning notation, throughout the present and the next chapters in Part I we make use of the following conventions. The set $\mathsf{Var} = \{p, q, \dots, \}$ denotes a set of propositional variables; strong negation is denoted $\sim p$, for a propositional variable $p \in \mathsf{Var}$. Given two sets X, Y we denote the set-theoretic difference as $X \setminus Y$ and the Cartesian product of X and Y as $X \times Y$. Sequences are denoted $\langle x_0, \dots, x_n \rangle$ or $[x_0, \dots, x_n]$. Given a sequence $\vec{x} = \langle x_0, \dots, x_n \rangle$ and an element x, we denote by $\vec{x}^\frown\langle x \rangle$ the concatenation of $\vec{x}$ with x, i.e. the sequence $\langle x_0, \dots, x_n, x \rangle$ or $[x_0, \dots, x_n, x]$. If f is a function $f : X \to Y$ and $X' \subseteq X$, we define $f[X'] = \{f(a) \in Y \mid a \in X'\}$. Given a family of sets $\mathbf{M}$, its union is denoted $\bigcup \mathbf{M}$.

[1] This criterion captures the preference for e.g. {*penguins do not fly*} over {*penguins are birds*, *birds fly*} in evidence-based reasoning, not considered here.

[2] More direct rules can fail to detect interactions. Consider, for instance, two moving objects that are directed against each other. Under non-detailed rules, these objects would *magically* not collide but reach their destinations untroubled.

After fixing the basic notation used henceforth, let us describe with more detail the language of t-DeLP and some representational issues relevant to the study of argumentation-theoretic properties [35] in later sections.

The language of t-DeLP builds upon a set of temporal literals, consisting of a pair $\langle$*literal, time*$\rangle$. Literals are expressions of the form p or $\sim p$ from the set of variables $p \in \mathsf{Var}$. Strong negation $\sim$ cannot be nested, so we will use the following notation over literals: if $\ell = p$ then $\sim\ell$ will denote $\sim p$, and if $\ell = \sim p$ then $\sim\ell$ will denote p. More generally, we also use $\sim\langle p, t\rangle$ to denote $\langle\sim p, t\rangle$; the set of negations of literals in a set X is denoted $\sim X = \{\sim\ell \mid \ell \in X\}$. Although we will only refer to propositional variables and literals throughout the next chapters, these might rather be seen as ground predicates, e.g. of the form *literal* = *property*(*object*) or *literal* = *parameter*(*object*, *value*). In the same direction, we will later introduce general rules (or rule schemas) as in logic programming.

Temporal parameters in t-DeLP will take discrete values in $\mathbb{N}$ and will be denoted with t or expressions t_i with subindexes. Thus, a temporal literal is of the form $\langle\ell, t\rangle$. Time is relevant to determine whether a pair of temporal literals contradict each other: for such a contradiction to exist, the literals expressed must be the negation of each other*and* they must be claimed to hold at the same time: $\langle\ell, t\rangle$ and $\langle\sim\ell, t\rangle$ are contradictory. A (domain-specific) temporal or causal rule is an expression of the form

a temporal literal		a set of temporal literals	
$\mathsf{head}(\delta)$		$\mathsf{body}(\delta)$	
$\langle\ell, t\rangle$	$\leftarrow$	$\langle\ell_1, t_1\rangle, \ldots, \langle\ell_n, t_n\rangle$	strict rule
$\langle\ell, t\rangle$	$\prec$	$\langle\ell_1, t_1\rangle, \ldots, \langle\ell_n, t_n\rangle$	defeasible rule

A strict rule represents a sound inference step, while a defeasible rule captures a reasonable but possibly unsound inference step. In t-DeLP, these can be read as follows:

(strict)	if the body is true, then the head is true
(defeasible)	if the body is true, then the head is true, unless good enough reasons exist to the contrary

Different types of such rules deserve explicit mentioning.

- temporal facts $\langle\ell, t\rangle \leftarrow$ are strict rules with an empty body, although for simplicity they are represented just by the temporal literal $\langle\ell, t\rangle$. (Not to be confused with a strictly or defeasibly derived temporal literal, also denoted $\langle\ell, t\rangle$; only the former can occur in the base of an argument.)
- persistence rules $\langle\ell, t+1\rangle \prec\langle\ell, t\rangle$ are defeasible rules stating that the truth of a literal ℓ is preserved from time t to the next time point $t+1$
- static rules $\langle\ell, t\rangle \leftarrow \langle\ell_1, t\rangle, \ldots, \langle\ell_n, t\rangle$, or similarly with $\prec$, are about a unique time point t, and hence represent constraints within this time point; mutex rules (see below) are a particular case of strict static rules.

In the next Section 1.3, we present a general notion of t-DeLP logical consequence, a relation between arbitrary logical programs (sets of facts and rules) and temporal literals (queries). In later Sections 1.4 we study this logical consequence for two sub-classes of t-DeLP programs, called simple and mutex. Simple programs contain no strict rules, so they just consist of a set of strict temporal facts and a set of defeasible rules. Thus, the only static "strict" constraints in simple programs are those of the form $\langle p, t\rangle$ and $\langle \sim p, t\rangle$.

This might prove insufficient when reasoning with expressions of the form

$$\langle \mathit{parameter}(\mathit{object}, \mathit{value}), \mathit{time}\rangle \quad \text{represented as} \quad \begin{cases} \langle p(o, v), t\rangle, \\ \text{or also } \langle p_v(o), t\rangle \\ \text{or even } \langle v(o), t\rangle. \end{cases}$$

For example, an object cannot have different values of a given parameter at a given time, so a strict incompatibility should exist between $\mathit{red}(\mathit{apple})$ and $\mathit{green}(\mathit{apple})$, and more generally for any pair $\langle p(o, v), t\rangle$ and $\langle p(o, v'), t\rangle$ of expressions as above if $v \neq v'$. In other cases, two objects $o \neq o'$ cannot share the same value at the same time, e.g. for spatial location.

These kinds of constraints have also been studied in the planning tradition, where they are called *mutex*, for mutual exclusion. A mutex constraint between pairs of the form $\langle p(o, v), t\rangle$ and $\langle p(o, v'), t\rangle$ (with t arbitrary) is expressed, in planning, by a joint membership in some set $X = \{p(o, v), p(o, v'), \ldots\}$, the latter called a mutex set.

In t-DeLP, though, mutex constraints are represented by strict rules, e.g. $\langle \sim p(o, v'), t\rangle \leftarrow \langle p(o, v)\rangle$. The logic and planning representations of mutex constraints can easily be induced from each other.

Example 1.2.1. Let $\mathcal{O}$ and $\mathcal{L}$ be the sets of objects o and locations l; and let $@(o, l) \in \mathsf{Var}$ denote: *object o is at location l*;

- the *at most one location per object* policy is defined by a mutex set $X_o = \{o\} \times \mathcal{L}$, for each $o \in \mathcal{O}$; this set X_o corresponds to the set of rules of the form

$$\langle \sim @(o, l), t\rangle \leftarrow \langle @(o, l'), t\rangle \qquad \text{for each } l, l' \in \mathcal{L} \text{ with } l \neq l'$$

- the *at most one object per location* policy is defined by a mutex set $X_l = \mathcal{O} \times \{l\}$, for each $l \in \mathcal{L}$; this set X_l corresponds to the set of rules

$$\langle \sim @(o, l), t\rangle \leftarrow \langle @(o', l), t\rangle \qquad \text{for each } o, o' \in \mathcal{O} \text{ with } o \neq o'$$

1.3 A general t-DeLP framework

In a sketch, argumentation-based logic programming formalisms work as follows: we start with a knowledge base, a logical program (Π, Δ) with temporal facts and rules, and a query $\langle \ell, t\rangle$; we combine facts and rules in (Π, Δ) into an

argument $\mathcal{A}$ for the query, i.e. an argument whose conclusion is the query. This argument is a set $\mathcal{A} \subseteq \Pi \cup \Delta$ that entails the (presumable) fact $\langle \ell, t \rangle$ by applying *modus ponens*; this set $\mathcal{A}$ must be $\subseteq$-minimal with this property. Once some such argument $\mathcal{A}$ for $\langle \ell, t \rangle$ is fixed, an argumentative process in (Π, Δ) generates counter-arguments $\mathcal{B}$ that defeat $\mathcal{A}$; that is, arguments $\mathcal{B}$ whose conclusions contradict some non-basic element of $\mathsf{literals}(\mathcal{A})$ –i.e. arguments $\mathcal{B}$ attacking $\mathcal{A}$– and satisfying some extra conditions for defeat. Then arguments $\mathcal{C}$ defending $\mathcal{A}$ by way of attacking some such $\mathcal{B}$ are considered too, and so on, until all the relevant arguments for and against are generated. These arguments can be arranged in the form of a tree that has $\mathcal{A}$ as its root, and its arcs are the defeat relation, and so terminal nodes are arguments for which no defeaters exist. At this point, $\mathcal{A}$ is assigned a label (undefeated, or defeated), according to a recursive labeling procedure in this tree of arguments. The procedure determines whether $\mathcal{A}$ is undefeated, i.e. whether it constitutes a solid justification or explanation for the truth of $\langle \ell, t \rangle$. In case it is, we say $\langle \ell, t \rangle$ is warranted in the knowledge base (Π, Δ).

For the temporal component, we take the set of natural numbers $\mathbb{N}$ as our working set of discrete time points. The logic t-DeLP is based on temporal literals $\langle \ell, t \rangle$, where ℓ is a literal and $t \in \mathbb{N}$, denoting ℓ *holds at time* t. In order to solve conflicts between arguments, the preference (or defeat) relation between arguments will be based on: a preference for arguments with *more premises* and *more recent* information. The latter criterion denotes a preference for arguments claiming a change (say from $\langle \sim\ell, t \rangle$ to $\langle \sim\ell, t+1 \rangle$) over arguments based on the persistence of ℓ from t to $t+1$, if the sub-arguments for $\langle \ell, t \rangle$ contained in $\mathcal{A}$ and $\mathcal{B}$ are the identical. In addition, since arguments must be consistent with strict information, strict arguments cannot be attacked.

Definition 1.3.1 (Literal, Rule)**.** Given a finite set of propositional variables Var, we define $\mathsf{Lit} = \mathsf{Var} \cup \{\sim p \mid p \in \mathsf{Var}\}$. The set of temporal literals is defined as $\mathsf{TLit} = \{\langle \ell, t \rangle \mid \ell \in \mathsf{Lit}, t \in \mathbb{N}\}$. A *temporal defeasible (resp. strict) rule* is an expression δ relating temporal literals of the form

$$\langle \ell, t \rangle \ \multimap \langle \ell_0, t_0 \rangle, \ldots, \langle \ell_n, t_n \rangle \qquad (\text{resp.} \quad \langle \ell, t \rangle \leftarrow \langle \ell_0, t_0 \rangle, \ldots, \langle \ell_n, t_n \rangle),$$

where $t \geq \max\{t_0, \ldots t_n\}$. We write $\mathsf{body}(\delta) = \{\langle \ell_0, t_0 \rangle, \ldots, \langle \ell_n, t_n \rangle\}$, $\mathsf{head}(\delta) = \langle \ell, t \rangle$ and $\mathsf{literals}(\delta) = \{\mathsf{head}(\delta)\} \cup \mathsf{body}(\delta)$.

As we mentioned, a strict rule with an empty body $\langle \ell, t \rangle \leftarrow$ represents a basic fact that holds at time t. As in most of the DeLP literature basic defeasible facts of the form $\langle \ell, t \rangle \multimap$, also called presumptions in [36], are not considered. The set of (defeasible) persistence rules $\langle \ell, t+1 \rangle \multimap \langle \ell, t \rangle$ will be denoted Δ_{p}. In contrast, strict persistence rules and –more generally– strict durative rules carry such a strong commitment on the preservation of a fact or its future occurrence, that they will not be considered.

Definition 1.3.2 (Derivability, Consistent Set)**.** Given a set of rules and strict facts Γ, we say a literal $\langle \ell, t \rangle$ *derives from* Γ, denoted $\Gamma \vdash \langle \ell, t \rangle$ or also $\langle \ell, t \rangle \in \mathrm{Cn}(\Gamma)$ iff

- $\langle \ell, t \rangle \in \Gamma$, or
- there exists $\delta \in \Gamma$ with $\mathsf{head}(\delta) = \langle \ell, t \rangle$, and such that $\mathsf{body}(\delta)$ is a set of literals that derive from Γ.

We say Γ is *consistent* iff the set $\mathrm{Cn}(\Gamma)$ contains no pair of literals of the form $\langle \ell, t \rangle$ and $\langle \sim\ell, t \rangle$. In particular, a set of literals is consistent iff it does not contain such a contradictory pair of literals $\langle \ell, t \rangle, \langle \sim\ell, t \rangle$.

Note that derivability is monotonic: $\mathrm{Cn}(\Gamma) \subseteq \mathrm{Cn}(\Gamma')$ whenever $\Gamma \subseteq \Gamma'$.

Definition 1.3.3 (Program)**.** A *t-DeLP program* is a pair (Π, Δ) where $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathsf{r}}$ is a consistent set of temporal strict facts Π_{f} and rules Π_{r}, and Δ is a set of temporal defeasible rules.

Temporal rules as above can be seen as instances of general rules δ^* of the form

$$\ell \ -\!\prec (\ell_0, d_0), \ldots, (\ell_n, d_n)$$

–and similarly for strict rules with the $\leftarrow$ symbol–, where each d_i expresses how much time in advance must ℓ_i hold for the rule to apply and produce a derivation of ℓ. Such a general rule δ^* is to be understood as a shorthand for the set of rules

$$\{\langle \ell, t \rangle \ -\!\prec \langle \ell_0, t - d_0 \rangle, \ldots, \langle \ell_n, t - d_n \rangle \mid t \in \mathbb{N}, t \geq \max\{d_0, \ldots, d_n\}\}.$$

For example, the rule

$$\langle p, 4 \rangle \ -\!\prec \langle q, 3 \rangle \text{ would be an instance of } p \ -\!\prec (q, 1).$$

Persistence rules can therefore be expressed as general rules of the form $\ell -\!\prec (\ell, 1)$; this defeasible general persistence rule for ℓ will be denoted δ_ℓ, and an instance $\langle \ell, t+1 \rangle -\!\prec \langle \ell, t \rangle$ of δ_ℓ will also be denoted by $\delta_\ell(t)$; similarly, the set of δ_ℓ-instances in the interval $[t, \ldots, t+k]$ will be denoted by $\{\delta_\ell(t')\}_{t \leq t' \leq t+k}$.

Though the notation for general rules becomes handy in some examples, the formal definitions below do make use only of instantiated temporal rules. Unless stated otherwise, in the remaining of the chapter we will mean by *rule* an expression as in Definition 1.3.1 that has not a non-empty body.

Example 1.3.4 (Snake Bite)**.** Consider the situation described next and formalized in Figure 1.3.4. Lars, a tourist visiting the Snake Forest, has just been bitten by a venomous snake. These two facts are denoted @forest(Lars) and resp. bitten^*(Lars). [3] The poison of this type of snake does kill a person in 3 hours (δ_1). But since our subject, Lars, is experienced (it has been bitten and cured a few times before), denoted exp(Lars), he may resist up to 5 hours (δ_2, δ_3). We decide to take him to the nearest hospital. In normal conditions this would take

[3] We use two literals $\mathsf{bitten}^*(\cdot)$ and $\mathsf{bitten}(\cdot)$. The literal with an asterisk is used to track the (unique) time where the snake bite occurred, and hence will not be allowed to persist (i.e. no persistence rules for this literal will exist in the program). The second literal $\mathsf{bitten}(\cdot)$ just denotes the fact of *having been (recently) bitten* and persistence rules for it are assumed.

2 hours (δ_4), but since today is sunday, the traffic jam (δ_7) makes it impossible to reach the hospital in less than 4 hours (δ_5, δ_6). The antidote takes less than an hour to become effective (δ_8), and is given to persons that are at the hospital, have been recently bitten (denoted $\mathsf{bitten}(\cdot)$) and are alive (denoted $\sim\mathsf{dead}(\cdot)$). We prove below in t-DeLP that Lars survives the snake attack.

Π

$$\left\{\begin{array}{lll}\langle @\mathsf{forest}(\mathrm{Lars}), 0\rangle, & \langle \mathsf{bitten}^*(\mathrm{Lars}), 0\rangle, & \langle \mathsf{exp}(\mathrm{Lars}), 0\rangle, \\ \langle \sim\mathsf{dead}(\mathrm{Lars}), 0\rangle, & \langle \mathsf{sunday}, 0\rangle & \end{array}\right\}$$

Δ

$\mathsf{bitten}(\mathrm{Lars})$	$\prec$	$\langle \mathsf{bitten}^*(\mathrm{Lars}), 0\rangle$	δ_0
$\mathsf{dead}(\mathrm{Lars})$	$\prec$	$(\mathsf{bitten}^*(\mathrm{Lars}), 3)$	δ_1
$\sim\mathsf{dead}(\mathrm{Lars})$	$\prec$	$(\mathsf{bitten}^*(\mathrm{Lars}), 3), (\mathsf{exp}(\mathrm{Lars}), 3), (\sim\mathsf{dead}(\mathrm{Lars}), 3)$	δ_2
$\mathsf{dead}(\mathrm{Lars})$	$\prec$	$(\mathsf{bitten}^*(\mathrm{Lars}), 5), (\mathsf{exp}(\mathrm{Lars}), 5), (\sim\mathsf{dead}(\mathrm{Lars}), 5)$	δ_3
$@\mathsf{hospital}(\mathrm{Lars})$	$\prec$	$(\mathsf{bitten}^*(\mathrm{Lars}), 2), (@\mathsf{forest}(\mathrm{Lars}), 2), (\sim\mathsf{dead}(\mathrm{Lars}), 2)$	δ_4
$\sim@\mathsf{hospital}(\mathrm{Lars})$	$\prec$	$\left\{\begin{array}{ll}(\mathsf{traffic.jam}, 2), & (\mathsf{bitten}^*(\mathrm{Lars}), 2), \\ (\sim\mathsf{dead}(\mathrm{Lars}), 2), & (@\mathsf{forest}(\mathrm{Lars}), 2)\end{array}\right\}$	δ_5
$@\mathsf{hospital}(\mathrm{Lars})$	$\prec$	$\left\{\begin{array}{ll}(\mathsf{traffic.jam}, 4), & (\mathsf{bitten}^*(\mathrm{Lars}), 4), \\ (\sim\mathsf{dead}(\mathrm{Lars}), 4), & (@\mathsf{forest}(\mathrm{Lars}), 4)\end{array}\right\}$	δ_6
$\mathsf{traffic.jam}$	$\prec$	$(\mathsf{sunday}, 0)$	δ_7
$\sim\mathsf{dead}(\mathrm{Lars})$	$\prec$	$(@\mathsf{hospital}(\mathrm{Lars}), 1), (\mathsf{bitten}(\mathrm{Lars}), 1), (\sim\mathsf{dead}(\mathrm{Lars}), 1)$	δ_8
		plus $\delta_\ell \in \Delta_\mathsf{p}$ for each $\ell \notin \{\mathsf{bitten}^*(\mathrm{Lars}), \sim@\mathsf{loc}(\mathrm{Lars})\}$	δ_ℓ

Figure 1.1: The list of strict facts, defeasible rules δ_1-δ_8 and persistence rules δ_ℓ for Example 1.3.4.

As it happens in DeLP, the set of derivable literals in (Π, Δ) will not in general be consistent. The first step to recover consistency is to focus on those derivations that have the form of an argument.

Definition 1.3.5 (Argument)**.** Given a t-DeLP program (Π, Δ), an *argument* for $\langle \ell, t\rangle$ is a set $\mathcal{A} = \mathcal{A}_\Pi \cup \mathcal{A}_\Delta$, with $\mathcal{A}_\Pi \subseteq \Pi$ and $\mathcal{A}_\Delta \subseteq \Delta$, such that:

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

Thus, arguments are non-redundant derivations, consistent with the strict part of the program, and which make use of defeasible information only when strict information is not available. In particular, if a strict argument exists for some literal, then no defeasible derivation for the same literal constitutes an argument.

In Example 1.3.4, each possible argument consists of facts in Π_{f} and rules in Δ. Observe that, although Π and Δ may be infinite (due to the coding of general rules as an infinite set of temporal rules), an argument for a program (Π, Δ) will always be a finite subset of $\Pi \cup \Delta$. Given an argument $\mathcal{A}$ for $\langle \ell, t \rangle$, we also define:

$$\begin{aligned} \mathsf{concl}(\mathcal{A}) &= \langle \ell, t \rangle \\ \mathsf{base}(\mathcal{A}) &= \mathsf{body}[\mathcal{A}] \setminus \mathsf{head}[\mathcal{A} \setminus \Pi_{\mathsf{f}}] \\ \mathsf{literals}(\mathcal{A}) &= (\bigcup \mathsf{body}[\mathcal{A}]) \cup \mathsf{head}[\mathcal{A}] \end{aligned}$$

Note the definition of $\mathsf{base}(\cdot)$ applies to arbitrary sets of facts and rules $\mathcal{A} \subseteq \Pi \cup \Delta$, not only arguments. For the particular case of arguments, a simpler characterization is possible.

Fact 1.3.6. If $\mathcal{A}$ is an argument in (Π, Δ), then $\mathsf{base}(\mathcal{A}) = \mathcal{A} \cap \Pi_{\mathsf{f}}$.

Similarly, the conclusion of an argument $\mathcal{A}$ can also be characterized as the only head of a rule in $\mathcal{A}$ which is not used by other rules in $\mathcal{A}$ to infer further literals, i.e. $\mathsf{concl}(\mathcal{A}) \notin \mathsf{body}[\mathcal{A}]$.

Proposition 1.3.7. *Let* (Π, Δ) *be a t-DeLP program, and let* $\mathcal{A}$ *be an argument for some* $\langle \ell, t \rangle = \mathsf{concl}(\mathcal{A})$. *Then* $\{\langle \ell, t \rangle\} = \mathsf{head}[\mathcal{A}] \setminus \bigcup \mathsf{body}[\mathcal{A}]$.

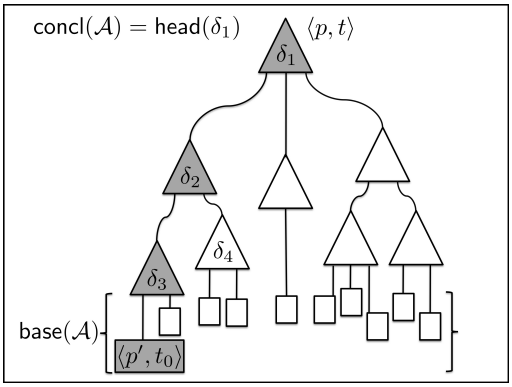


Figure 1.2: Facts from Π are represented as rectangles; and defeasible rules as triangles. The total duration of an argument $\mathcal{A}$ is the maximum sum of the durations of rules among paths from the conclusion to the base. In the Figure, this path consists of $\delta_1, \delta_2, \delta_3$ with a total duration of $t - t_0$.

Remark 1.3.8. In DeLP, arguments are defined as sets of defeasible rules $\mathcal{A} \subseteq \Delta$, leaving open how these are to be completed by Π to obtain a (minimal, consistent) derivation of some literal ℓ; since different completions in particular allow for different conclusions, one must make explicit which is the intended conclusion in the form (argument, conclusion). Thus, the DeLP notation for an argument is $\langle \mathcal{A}, \ell \rangle$. In contrast, we explicitly fix the strict rules in the definition

of an argument $\mathcal{A}$, so the conclusion $\mathsf{concl}(\mathcal{A})$ is uniquely determined by $\mathcal{A}$. The latter definition simplifies the detection of inconsistencies with intermediate steps in the strict part of $\mathcal{A}$. With more detail, there can be several ways to complete defeasible rules in $\mathcal{A}$ into a derivation for $\mathsf{concl}(\mathcal{A})$, and each of them can be attacked by different arguments. For example, the sets

$$\left\{\begin{array}{c}\langle p,4\rangle \leftarrow \langle q,2\rangle \\ \langle q,2\rangle \leftarrow \langle r,1\rangle \\ \langle r',0\rangle\end{array}\right\} \quad \text{and} \quad \left\{\begin{array}{c}\langle p,4\rangle \leftarrow \langle s,3\rangle \\ \langle s,3\rangle \leftarrow \langle r,1\rangle \\ \langle r',0\rangle\end{array}\right\}$$

may both complete the set of defeasible rules $\{\langle p',5\rangle \prec \langle p,4\rangle, \langle r,1\rangle \prec \langle r',0\rangle\} \subseteq \Delta$ into an argument (derivation) for $\langle p',5\rangle$, but only the completion on the right is attacked by an argument concluding $\langle \sim s,3\rangle$.

Now we proceed to define a sub-argument of an argument $\mathcal{A}$. A sub-argument will be the actual target of an attack by another argument.

Definition 1.3.9 (Sub-argument)**.** Let (Π,Δ) be a t-de.l.p. and let $\mathcal{A}$ be an argument for $\langle \ell,t\rangle$ in (Π,Δ). Given some $\langle \ell_0,t_0\rangle \in \mathsf{literals}(\mathcal{A})$, a *sub-argument* for $\langle \ell_0,t_0\rangle$ is a subset $\mathcal{B} \subseteq \mathcal{A}$ such that $\mathcal{B}$ is an argument for $\langle \ell_0,t_0\rangle$.

For example, in Figure 1.2, $\mathcal{A}(\mathsf{head}(\delta_2)) = \{\delta_2,\delta_3,\delta_4,\langle \ell',t'\rangle,\ldots\}$. The inductive definition for computing the sub-argument induced by some literal is straightforward (see Appendix 1.8).

Proposition 1.3.10. *Given some argument $\mathcal{A}$ and a literal $\langle \ell,t\rangle \in \mathsf{literals}(\mathcal{A})$, then the sub-argument of $\mathcal{A}$ for $\langle \ell,t\rangle$ is unique.*

From here on, this unique sub-argument of $\mathcal{A}$ induced by $\langle \ell_0,t_0\rangle$ will be denoted $\mathcal{A}(\langle \ell_0,t_0\rangle)$.

Definition 1.3.11 (Attack)**.** Given a t-DeLP program (Π,Δ), let $\mathcal{A}_0$ and $\mathcal{A}_1$ be arguments. We say $\mathcal{A}_1$ *attacks* $\mathcal{A}_0$ iff $\sim\mathsf{concl}(\mathcal{A}_1) \in \mathsf{literals}(\mathcal{A}_0)$. In this case, we also say that $\mathcal{A}_1$ attacks $\mathcal{A}_0$ at the sub-argument $\mathcal{A}_0(\sim\mathsf{concl}(\mathcal{A}_1))$.

Notice that an argument $\mathcal{A}_1$ cannot attack another $\mathcal{A}_0$ at a sub-argument consisting of strict information only (i.e. if $\mathcal{A}_0(\sim\mathsf{concl}(\mathcal{A}_1)) \subseteq \Pi$), since in this case $\mathcal{A}_1$ would not be consistent with Π, and hence $\mathcal{A}_1$ would not even be an argument.

As in DeLP, one refines the relation of attack relation into a defeat relation to decide which argument prevails in case of an attack. This relation could be in principle specified by the user[4], but in this and the next chapter we adopt a new formal criterion meeting the intuitive preferences exemplified next.

Example 1.3.12 (Snake Bite; cont'd)**.** See Figure 1.3 for an illustration of Example 1.3.4. The arguments are defined by the following rules (facts are not listed here):

[4]See [61] for an alternative procedure based on a preference relation between rules.

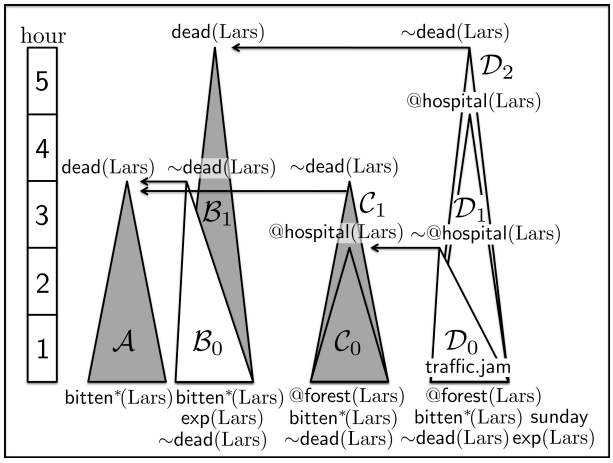


Figure 1.3: The Snake Bite scenario. Arguments are depicted as triangles, with arrows denoting conflicts among them. Arguments for which defeaters exist are depicted in grey. (In this example, the arguments in grey are also defeated arguments.)

$$\mathcal{A} \supseteq \{\delta_1(0)\} \qquad \mathcal{B}_0 \supseteq \{\delta_2(0)\} \qquad \mathcal{B}_1 \supseteq \{\delta_3(0)\} \qquad \mathcal{C}_0 \supseteq \{\delta_4(0)\}$$
$$\mathcal{C}_1 \supseteq \{\delta_4(0), \delta_8(2)\} \cup \{\delta_0(t), \delta_{\mathsf{bitten}(\mathrm{Lars})}(t), \delta_{\sim\mathsf{dead}(\mathrm{Lars})}(t)\}_{0\leq t<2}$$
$$\mathcal{D}_0 \supseteq \{\delta_7(0), \delta_5(0)\}$$
$$\mathcal{D}_1 \supseteq \mathcal{B}_0 \cup \{\delta_{\sim\mathsf{dead}(\mathrm{Lars})}(3), \delta_7(0), \delta_6(0)\}$$
$$\mathcal{D}_2 \supseteq \mathcal{D}_1 \cup \{\delta_0(0), \delta_8(4)\} \cup \{\delta_{\mathsf{bitten}(\mathrm{Lars})}(t)\}_{0\leq t\leq 3}$$

The arguments related by an arrow attack each other: $\mathcal{C}_1$, $\mathcal{B}_0$ attack $\mathcal{A}$ and viceversa. But there are asymmetries in the quantity of information supporting each argument. Intuitively, in this example we have that

> $\mathcal{B}_0$ should prevail over $\mathcal{A}$ since it is based on more strict facts (the premises of $\mathcal{A}$ are a proper subset of those in $\mathcal{B}_0$); such an asymmetry between $\mathcal{B}_0$ and $\mathcal{A}$ makes the latter not to count as a reason against $\mathcal{B}_0$. (See also Figure 1.3.)

To illustrate another kind of asymmetry in the quality of information between arguments, consider a new example:

Example 1.3.13 (Falling object)**.** Suppose you hold an object o at some distance d_0 from the floor, and drop it at $t = 0$. It is expected to crash into the floor at, say, $t = 3$. This is modeled by an argument $\mathcal{A}$ having $\mathsf{base}(\mathcal{A}) = \{\langle @(o, d_0), 0\rangle\}$, intermediate steps $\langle @(o, d_1), 1\rangle$ and $\langle @(o, d_2), 2\rangle$ (i.e. both in $\mathsf{literals}(\mathcal{A})$ derived using appropriate rules), and conclusion $\mathsf{concl}(\mathcal{A}) = \langle @(o, 0), 3\rangle$; this latter literal $@(o, 0)$ denotes *o is at the floor*. Now, admitting (as we do) persistence rules for positive facts like $@(o, \cdot)$, an argument $\mathcal{B}$ can be constructed for the conclusion that the object will remain floating over the floor

at d_1. Namely, let $\mathcal{B} = \mathcal{A}(\langle @(o, d_1), 1\rangle) \cup \{\delta(t)_{@(o,d_1)}\}_{1 \leq t < 3}$. Note that while the asymmetry from Ex. 1.3.12 above is missing, since now $\mathsf{base}(\mathcal{B}) = \mathsf{base}(\mathcal{A})$,

> the argument $\mathcal{B}$ should not be a defeater for $\mathcal{A}$; the idea is that the existence of some reason for a change (like gravity, for heavy objects not on the ground) should override the use of persistence.

Otherwise, when o lies at the floor, the persistence of this fact seems a reasonable inference as long as no further changes are known of.

Finally, to illustrate blocking defeaters let us consider a rephrasing of the Snake Bite scenario in Example 1.3.4-1.3.12.

Example 1.3.14 (Snake Bite; cont'd)**.** Let us rewrite Example 1.3.4 with these rules: black-spotted snakes are generally poisonous, while green snakes are generally harmless.

> If *a green black-spotted snake bites Lars*, we are not able to decide whether he has been poisoned, since reasons for and against do not dominate each other.

The preferences from Examples 1.3.12-1.3.14 are formally captured by the general definition of defeat given next. These examples suggest two ways in which an argument $\mathcal{A}_1$ should defeat another argument $\mathcal{A}_0$, always comparing the former $\mathcal{A}_1$ with the sub-argument $\mathcal{B} \subseteq \mathcal{A}_0$ of the latter attacked by $\mathcal{A}_1$.

The first possibility is that the defeater argument $\mathcal{A}_1$ is based on more strict facts than the defeated $\mathcal{B}$; this is done with a set-theoretic comparison of the bases of each argument. The second possibility is based on a hierarchy of rules in terms of relative strength (denoted $>$):

$$\begin{array}{rcccl} \text{strict rules} & > & \begin{array}{c}\text{non-persistence}\\ \text{defeasible rules}\end{array} & > & \text{persistence rules} \\ \Pi_{\mathsf{r}} & > & \Delta \smallsetminus \Delta_{\mathsf{p}} & > & \Delta_{\mathsf{p}} \end{array}$$

As later shown, this second criteria aims to require the defeater $\mathcal{A}_1$ to make less use of persistence rules than the defeated $\mathcal{B}$. This criterion is formalized as follows: first, we identify the maximal sub-argument (if unique) that occurs in both arguments $\mathcal{A}_1 \cap \mathcal{B}$; then, the remaining part of the defeater $\mathcal{A}_1$ must consist of some non-persistence rules, while the remaining of the defeated $\mathcal{B}$ must consist of persistence rules (plus possibly some strict rules).

Definition 1.3.15 (Defeat)**.** Let (Π, Δ) be a t-DeLP program, and $\mathcal{A}_0, \mathcal{A}_1$ arguments such that $\mathcal{A}_1$ attacks $\mathcal{A}_0$ at $\mathcal{B}$; say $\mathsf{concl}(\mathcal{B}) = \langle \ell, t\rangle$ and $\mathsf{concl}(\mathcal{A}_1) = \langle \sim\ell, t\rangle$. We say $\mathcal{A}_1$ is a *proper defeater for* $\mathcal{A}_0$, denoted $\mathcal{A}_1 \succ \mathcal{A}_0$, iff

- $\mathsf{base}(\mathcal{A}_1) \supsetneq \mathsf{base}(\mathcal{B})$, or
- first, $\mathcal{A}_1 \cap \mathcal{B}$ is an argument for some $\langle \ell^*, t'\rangle$ with $t' < t$; second, $\mathcal{B} \smallsetminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi$; third, $(\mathcal{B} \smallsetminus \mathcal{A}_1) \cap \Delta_{\mathsf{p}} \neq \varnothing$; fourth, $(\mathcal{A}_1 \smallsetminus \mathcal{B}) \cap (\Delta \smallsetminus \Delta_{\mathsf{p}}) \neq \varnothing$; and fifth, $\mathsf{base}(\mathcal{A}_1) \subseteq \mathsf{base}(\mathcal{B})$ implies $\mathsf{base}(\mathcal{A}_1) = \mathsf{base}(\mathcal{B})$.

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

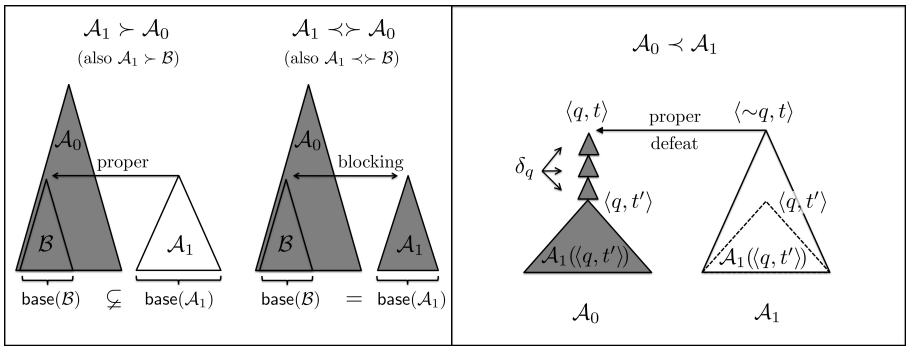


Figure 1.4: (Left) An illustration of the first criteria for proper and blocking defeat, based on a comparison between bases. (Right) A simple example of the second criteria for proper defeat, based on the use of persistence rules, e.g. δ_q.

In case an argument $\mathcal{A}_1$ attacking $\mathcal{A}_0$ is preferred to the latter after comparing their use of persistence rules (i.e. the second criterion in Def. 1.3.15), we will say that $\mathcal{A}_1$ is (informationally) longer than $\mathcal{A}_0$, since the attacked sub-argument of $\mathcal{A}_0$ is a fragment of $\mathcal{A}_1$ merely extended with persistence (and possibly strict rules).

Note that in the general case of t-DeLP programs, Definition 1.3.15 might be too restrictive: it will not apply to the case when the intersection $\mathcal{A}_1 \cap \mathcal{B}$, rather than being an argument, consists of two or more sub-arguments. The above Definition 1.3.15, adopted for the sake of simplicity, easily generalizes to the case where multiple sub-arguments are shared between $\mathcal{A}_1$ and $\mathcal{B}$. The above general definition of defeat is well-defined, is shown as follows.

Proposition 1.3.16. *The following hold for any t-DeLP program:*

(1) If $\mathcal{A}_1$ is a proper defeater for $\mathcal{A}_0$ at $\mathcal{B}$, then $\mathcal{B}$ is not a defeater for $\mathcal{A}_1$.

(2) If $\mathcal{A}, \mathcal{B}$ attack each other, and $\mathcal{B}$ is not a proper defeater for $\mathcal{A}$, then $\mathcal{A}$ is a defeater for $\mathcal{B}$.

For the particular cases of simple and mutex programs studied in the next sections, the general defeat relation in Definition 1.3.15 can be greatly simplified. For example, we will drop the fifth condition, which now is necessary to prove Proposition 1.3.16 (1). The next counter-example shows that this condition is necessary for this claim.

Example 1.3.17. Let us define the program (Π, Δ) as follows, where $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathsf{r}}$.

$$
\begin{array}{lcl}
\Pi_{\mathsf{f}} & = & \{\langle p, t_0\rangle, \langle q, t_0\rangle\} \\
\Pi_{\mathsf{r}} & = & \{\langle {\sim}s, t\rangle \leftarrow \langle q, t\rangle, \langle r, t\rangle\} \quad (= \delta_0) \\
\Delta & = & \Delta_{\mathsf{p}} \cup \left\{ \begin{array}{l} \langle r, t_1\rangle \multimap \langle p, t_0\rangle \\ \langle s, t_2\rangle \multimap \langle r, t_1\rangle \end{array} \right\} \begin{array}{l} (= \delta_1) \\ (= \delta_2) \end{array} \\
\Delta_{\mathsf{p}} & = & \{\delta_q(t_0), \delta_q(t_1), \delta_r(t_1)\}
\end{array}
$$

The arguments $\mathcal{A}_1$ and $\mathcal{B}$ defined next are constructible in this program.

$$\mathcal{A}_1 = \{\langle p, t_0\rangle, \delta_1, \delta_2\} \quad \text{and} \quad \mathcal{B} = \{\langle p, t_0\rangle, \langle q, t_0\rangle, \delta_1, \delta_q(t_0), \delta_q(t_1), \delta_0(t_2)\}$$

These arguments attack each other, since $\mathsf{concl}(\mathcal{A}_1) = \langle s, t\rangle = {\sim}\mathsf{concl}(\mathcal{B})$. Now, without the fifth condition from Def. 1.3.15, we might conclude both that $\mathcal{A}_1 \succ \mathcal{B}$ (due to the second criterion) and that $\mathcal{B} \succ \mathcal{A}_1$ due to $\mathsf{base}(\mathcal{B}) \supseteq \mathsf{base}(\mathcal{A}_1)$.

The two criteria for proper defeat in Definition 1.3.15 suit the informal requirements presented above in Examples 1.3.12 and 1.3.13. Example 1.3.14 is also captured by the definition of blocking defeater in Def. 1.3.15.

The general defeat relation considered for t-DeLP is slightly different from that originally proposed for DeLP. We refer the reader to Section 1.6.2 for a detailed comparison of the DeLP and t-DeLP criteria for defeat.

As we said, an argument $\mathcal{B}$ defeating $\mathcal{A}$ can at its turn have its own defeaters $\mathcal{C}, \ldots$ and so on. (This is the case of $\mathcal{A}, \mathcal{B}_0, \mathcal{C}_0$ in Figure 1.3.) This gives rise to *argumentation lines* where each argument defeats its predecessor. Argumentation lines, though, are not simply the composition of the defeat relation: again we refine this composition by imposing some further constraints. These constraints are needed to enforce desirable properties: finite length, acyclicity, and intuitive defense relations (counterattacks). For instance, in an argumentation line $[\ldots, \mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots]$ we exclude the case where $\mathcal{C}$ is a blocking defeater for $\mathcal{B}$, *provided that* $\mathcal{B}$ is already blocking defeater for $\mathcal{A}$. This prevents the case $[\ldots, \mathcal{A}, \mathcal{B}, \mathcal{A}, \ldots]$. Other forms of cyclic defeats $[\ldots, \mathcal{A}, \mathcal{B}, \ldots, \mathcal{A}, \mathcal{B}, \ldots]$ are also excluded in the definition. The following definition is adapted from [61] to the present framework.

Definition 1.3.18 (Argumentation Line, Dialectical Tree)**.** Let $\mathcal{A}_1$ be an argument in (Π, Δ). An *argumentation line* for $\mathcal{A}_1$ is a sequence $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots]$ where

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

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

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

An argumentation line $[\mathcal{A}_1, \ldots, \mathcal{A}_n]$ for $\mathcal{A}_1$ is *maximal* if there is no other argument $\mathcal{A}_{n+1}$ such that $[\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}]$ is an arg. line for $\mathcal{A}_1$.

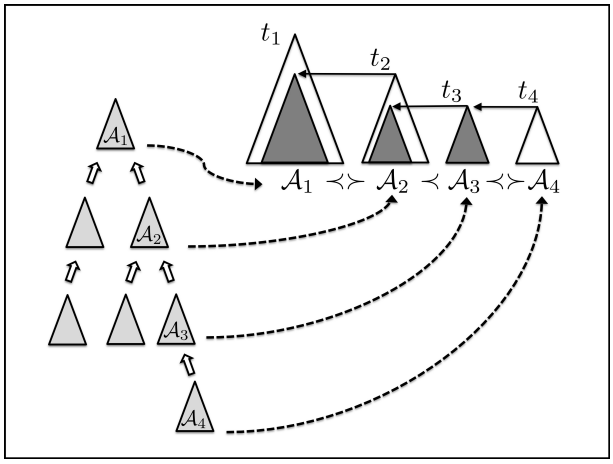


Figure 1.5: (Right) An argumentation line $\Lambda = [\mathcal{A}_1, \ldots, \mathcal{A}_4]$, with defeated sub-arguments depicted in grey. Notice that the time of these attacks is decreasing, and that condition (iii) from Def. 1.3.18 is satisfied. (Left) The same argumentation line Λ is depicted as part of the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$.

The set of argumentation lines for $\mathcal{A}_1$ can be arranged in the form of a tree, where all paths from the root $\mathcal{A}_1$ to the leaf nodes exactly correspond to all the possible maximal argumentation lines for $\mathcal{A}_1$. This tree is called *dialectical tree* for $\mathcal{A}_1$, and is denoted $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. We will also express that a sequence of arguments $\Lambda = [\mathcal{A}_1, \ldots]$ is a (non-necessarily maximal) argumentation line for $\mathcal{A}_1$ by $\Lambda \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$.

Remark 1.3.19. While (i) and (iii) are exactly as in DeLP, the above condition (ii) is less restrictive than its counterpart in [61]. In this work, a sub-argument of $\mathcal{A}_i$ cannot (indirectly) defend this argument. That is, a sub-argument of $\mathcal{A}_i$ cannot occur as $\mathcal{A}_{i+2j}$ in the same argumentation line. In our temporal case, we adopt a more liberal view concerning defenses based on sub-arguments: for instance, a sub-argument talking about a previous time might offer legitimate reasons to the defense of $\mathcal{A}_i$. If its only available defense was in this sense a proper part of the attacked argument, then it should be admitted. (See the next example.)

Example 1.3.20 (Falling object, cont'd)**.** We expand the scenario from Example 1.3.13 as follows: suppose as before that an object o will fall from height or distance d_0 into the floor, denoted by distance 0. That is, a transition from $\langle @(o, d_0), \cdot \rangle$ to $\langle @(o, 0), \cdot \rangle$ will happen. Moreover, assume that o is an egg, and also that a boiling pot of water is awaiting at the floor. The temperature at d_1 is cold (i.e. not hot). Thus, we have

$$\Pi_{\mathsf{f}} = \{\langle @(o, d_0), 0\rangle, \langle \mathsf{hot}(0), 0\rangle, \langle \sim\mathsf{hot}(d_1), 0\rangle, \langle \sim\mathsf{boils}(o), 0\rangle\}$$

The set Δ is as before in Example 1.3.12 (2), plus persistence rules for local heat or coldness: $\delta_{\mathsf{hot}(0)}, \delta_{\sim\mathsf{hot}(d_1)}$, and also the heat-transfer rules

$$\begin{array}{lrcl} \delta_1(t): & \langle \mathsf{boils}(o), t+1\rangle & \prec & \langle \mathsf{hot}(0), t\rangle, \langle @(o,0), t\rangle \\ \delta_2(t): & \langle \sim\mathsf{boils}(o), t+1\rangle & \prec & \langle \sim\mathsf{hot}(d_1), t\rangle, \langle @(o,d_1), t\rangle \end{array}$$

The arguments to conclude that the egg will ($\mathcal{A}^+$) or will not ($\mathcal{B}^+$, among others) boil at $t = 4$ are defined as

$$\begin{array}{rcl} \mathcal{A}^+ & = & \mathcal{A} \cup \{\langle \mathsf{hot}(0), 0\rangle\} \cup \{\delta(t)_{\mathsf{hot}(0)}\}_{0\le t<3} \cup \{\delta_1(3)\} \\ \mathcal{B}^+ & = & \mathcal{B} \cup \{\langle \sim\mathsf{hot}(d_1), 0\rangle\} \cup \{\delta(t)_{\sim\mathsf{hot}(d_1)}\}_{0\le t<3} \cup \{\delta_2(3)\} \end{array}$$

where $\mathcal{A}$ and $\mathcal{B}$ are as in Example 1.3.12 (2). We have $\mathsf{concl}(\mathcal{A}^+) = \langle \mathsf{boils}(o), 4\rangle$ and $\mathsf{concl}(\mathcal{B}^+) = \langle \sim\mathsf{boils}(o), 4\rangle$, so $\mathcal{A}^+$ and $\mathcal{B}^+$ attack each other, just like $\mathcal{A}$ and $\mathcal{B}$. But now, $\mathcal{A}^+$ represents as expected the fall-and-boiling of the egg while $\mathcal{B}^+$ states that the egg keeps floating in air and stays unboiled. The problem is that now $\mathcal{A}^+$ is not longer than $\mathcal{B}^+$ (in contrast to the previous arguments $\mathcal{A}$ and $\mathcal{B}$). In fact, we need $\mathcal{A}$ to defeat $\mathcal{B}^+$ at $\mathcal{B}$. Thus, Definition 1.3.18 allows for $[\mathcal{A}^+, \mathcal{B}^+, \mathcal{A}]$ to be an arg. line, so $\mathcal{A}$ can defend $\mathcal{A}^+$. If these defending sub-arguments were not allowed (see Definition 1.3.22 below), we could not conclude that the egg boils at $t = 4$ is a warranted conclusion.

Lemma 1.3.21. *For any t-DeLP program* (Π, Δ)*,*

(1) If $[\mathcal{A}_1, \ldots, \mathcal{A}_m, \ldots, \mathcal{A}_n]$ *is an argumentation line for* $\mathcal{A}_1$*, then* $[\mathcal{A}_m, \ldots, \mathcal{A}_n]$ *is an argumentation line for* $\mathcal{A}_m$*.*

(2) Each argumentation line $\Lambda = [\mathcal{A}_1, \ldots] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ *is finite. The dialectical tree* $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ *is finite.*

The following definitions of the marking procedure of dialectical trees and the notion of warrant follow exactly those of DeLP.

Definition 1.3.22 (Marking)**.** Let $\mathcal{T} = \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ be the dialectical tree for $\mathcal{A}_1$ in a given program (Π, Δ). Then,

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

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

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

Initially all the arguments in the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ are unmarked (grey) as in Figure 1.5 (Left). To illustrate the marking procedure, see Figure 1.6, where arguments marked U are represented white, and those marked D are represented black.

Note that in a dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$, an argument $\mathcal{A}$ can occur in different positions of several (maximal) argumentation lines in $\Lambda, \Lambda', \ldots \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. In this case, the marking of $\mathcal{A}$ in Λ can be different from the marking of $\mathcal{A}$ in Λ'. Given an argumentation line $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots, \mathcal{A}_n] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$, we will express the evaluation of its arguments along Λ according to the marking procedure by a corresponding sequence of D's and U's , e.g. $[D, D, U, \ldots, U]$.

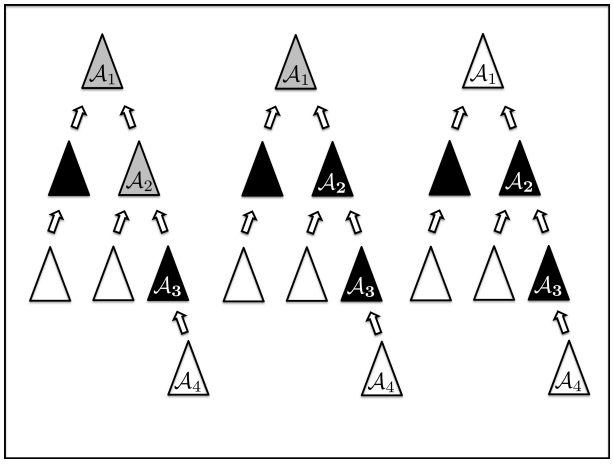


Figure 1.6: From left to right, the different steps for the marking procedure of arguments (triangles) in the dialectical tree for an argument $\mathcal{A}_1$: unmarked arguments are depicted in grey; arguments marked undefeated are depicted in white, and those marked defeated are depicted in black. In this case, the root argument is undefeated, so its conclusion is warranted.

Definition 1.3.23 (Warrant)**.** Given a t-DeLP program (Π, Δ), we say $\langle \ell, t \rangle$ is *warranted* in (Π, Δ) iff there exists an argument $\mathcal{A}_1$ for $\langle \ell, t \rangle$ in (Π, Δ) such that $\mathcal{A}_1$ is undefeated, i.e. marked U, in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. We will denote by $\mathsf{warr}(\Pi, \Delta)$ the set of warranted literals in (Π, Δ).

In the particular case of strict arguments $\mathcal{A} \subseteq \Pi$, we will have that $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$ only contains the argumentation line $[\mathcal{A}]$, so each strictly derivable fact is warranted. For any other argument $\mathcal{B}$, the strict argument $\mathcal{A}$ cannot occur in any argumentation line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$, since this would mean that the argument preceding (i.e. defeated by) $\mathcal{A}$ is actually not an argument.

Example 1.3.24 (Snake Bite; cont'd)**.** Recall Examples 1.3.4 and 1.3.12. The arguments in Fig. 1.3 related by an arrow stand in the relation of proper defeat, e.g. $\mathcal{A} \leftarrow \mathcal{B}_0$ denotes $\mathcal{B}_0$ is a proper defeater for $\mathcal{A}$. Thus we have the dialectical trees for each argument consist of the following argumentation lines (with the corresponding evaluations):

$$\begin{array}{rcll}
\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}) & = & \left\{\begin{array}{l}[\mathcal{A}, \mathcal{B}_0], \\ [\mathcal{A}, \mathcal{C}_1, \mathcal{D}_0]\end{array}\right\} & \begin{array}{l}[D, U], \\ [D, D, U]\end{array} \\
\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B}_0) & = & \{[\mathcal{B}_0]\} & [U] \\
\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B}_1) & = & \{[\mathcal{B}_1, \mathcal{D}_2]\} & [D, U] \\
\mathcal{T}_{(\Pi,\Delta)}(\mathcal{C}_i) & = & \{[\mathcal{C}_i, \mathcal{D}_0]\} & [D, U], \text{ for each } i \in \{0, 1\} \\
\mathcal{T}_{(\Pi,\Delta)}(\mathcal{D}_j) & = & \{[\mathcal{D}_j]\} & [U], \text{ for each } j \in \{0, 1, 2\}
\end{array}$$

Since $\mathcal{D}_2$ is undefeated, we (defeasibly) conclude that Lars will be alive at $t = 5$.

Example 1.3.25 (Falling object; cont'd)**.** Let us now solve Example 1.3.13. Recall the argument $\mathcal{A}$ concluding that dropping the object will indeed cause it to crash into the floor (distance 0) at $t = 3$. Now, $\mathcal{A}$ is a proper defeater for the rival arguments stating that the object will keep floating in the air once it reaches distance d_0, d_1 or d_2. Call these arguments $\mathcal{B}_0, \mathcal{B}_1$ and $\mathcal{B}_2$, resp. (Incidentally, note that $\mathcal{B}_1$ properly defeats $\mathcal{B}_0$ and so does $\mathcal{B}_2$ with $\mathcal{B}_0, \mathcal{B}_1$.) If these arguments capture all the relevant phenomena in this scenario, then $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}) = [\mathcal{A}]$ so $\mathcal{A}$ is undefeated; on the other hand, for any $0 \leq i < 3$ the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B}_i)$ contains a maximal argumentation line $\Lambda = [\mathcal{B}_i, \mathcal{A}]$, among possibly others. This can only be evaluated as $[D, U]$, so each $\mathcal{B}_i$ is defeated.

1.4 A study of t-DeLP for simple programs

After presenting the procedure for computing warrant in t-DeLP, we proceed to the logical study of t-DeLP, for certain sub-classes of programs. First we review the logical or argumentation-theoretical properties, called Rationality Postulates, studied in the present and the next chapter. Then we define the sub-class of simple programs and show that the restriction of t-DeLP to this class of programs implies that the rationality postulates are satisfied.

The Rationality Postulates were proposed by Caminada and Amgoud in [35] (see also [121]) to grant that certain types of counter-intuitive results do not occur in a given argumentation framework. (Compare the next definition with Def. **??**.)

Definition 1.4.1 (Rationality Postulates)**.** The Rationality Postulates, adapted to t-DeLP programs (Π, Δ), are as follows:

Sub-arguments:	if $\mathcal{A}$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$, then any sub-argument $\mathcal{A}'$ of $\mathcal{A}$ is also undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}')$.
Direct Consistency:	$\mathsf{warr}(\Pi, \Delta)$ is consistent.
Indirect Cons.:	$\mathsf{warr}(\Pi, \Delta) \cup \Pi$ (i.e. its logical closure) is consistent.
Closure:	$\mathrm{Cn}(\mathsf{warr}(\Pi, \Delta) \cup \Pi) \subseteq \mathsf{warr}(\Pi, \Delta)$ (strict consequences of warranted literals are warranted).

These postulates were discussed in [35] for some argumentation frameworks based on defeasible rule-based systems, and using any of the acceptability semantics proposed by Dung for abstract argumentation systems [52] (see Chapter C). We proceed to prove the Sub-arguments and Direct Consistency postulates in t-DeLP for simple programs. Neither Indirect Consistency nor Closure hold in general for arbitrary t-DeLP programs, as it happens in DeLP. However, these two postulates also hold for simple programs (they are trivial consequences of Direct Consistency for this class of programs). In the next section we will show that these four postulates are also satisfied by the sub-class of mutex programs.

Definition 1.4.2 (Simple program)**.** A t-DeLP *simple program* is a program (Π, Δ) containing no strict rules: $\Pi_{\mathsf{r}} = \varnothing$. In other words, $\Pi = \Pi_{\mathsf{f}}$.

For simple programs, lacking any strict rule, the relation of defeat instantiates as follows.

Definition 1.4.3 (Defeat for simple programs)**.** Let (Π, Δ) be a simple program, and $\mathcal{A}_0, \mathcal{A}_1, \mathcal{B}$ arguments such that $\mathcal{A}_1$ attacks $\mathcal{A}_0$ at $\mathcal{B}$ and $\mathsf{concl}(\mathcal{A}_1) = \langle \sim\ell, t\rangle$. We say $\mathcal{A}_1$ is a *proper defeater for* $\mathcal{A}_0$ iff

- $\mathsf{base}(\mathcal{A}_1) \supsetneq \mathsf{base}(\mathcal{B})$, or
- $\mathcal{B} = \mathcal{A}_1(\langle \ell, t'\rangle) \cup \{\delta_\ell(t'')\}_{t' \leq t'' < t}$, for some $t' < t$.

We say $\mathcal{A}_1$ is a *blocking defeater for* $\mathcal{A}_0$ when $\mathcal{A}_1$ attacks $\mathcal{A}_0$ but $\mathcal{A}_1 \not\succ \mathcal{A}_0$ and $\mathcal{A}_0 \not\succ \mathcal{A}_1$.

In other words, the second criterion for proper defeat (a less use of persistence) applies when the remaining (not shared) part of defeated argument is a set of persistence rules. Note that all the Examples in the previous Section 1.3 were expressed using simple programs. The first result states that the defeat relation is well-defined for simple programs.

Lemma 1.4.4. *For any simple program* (Π, Δ)*, Definitions 1.3.15 and 1.4.2 are equivalent.*

As a consequence from Proposition 1.3.16 and Lemma 1.4.4, we obtain that Definition 1.4.3 is well-defined.

Corollary 1.4.5. *The following hold for any simple t-DeLP program (under Def. 1.4.3 for defeat):*

(1) If $\mathcal{A}_1$ is a proper defeater for an argument $\mathcal{A}_0$ at $\mathcal{B}$, then $\mathcal{B}$ is not a defeater for $\mathcal{A}_1$.

(2) If $\mathcal{A}, \mathcal{B}$ attack each other, and $\mathcal{B}$ is not a proper defeater for $\mathcal{A}$, then $\mathcal{A}$ is a defeater for $\mathcal{B}$.

After these preliminaries, we proceed to prove the Rationality Postulates for simple programs. The next two results hold for t-DeLP programs in general, not just for simple programs. First we observe that "being marked defeated in a dialectical tree" can be expressed in the following more convenient form.

Remark 1.4.6. Let (Π, Δ) be a t-DeLP program, and let $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$ denote arguments in (Π, Δ). An argument $\mathcal{B}$ is marked defeated in an argumentation line $[\mathcal{A}, \ldots, \mathcal{B}] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$ iff there is an argument $\mathcal{C}$ marked undefeated in the argumentation line $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$.

Lemma 1.4.7. *Given a t-DeLP program* (Π, Δ)*, let $\mathcal{A}_1, \mathcal{A}_2$ be two arguments such that $\mathcal{A}_2$ is a defeater for $\mathcal{A}_1$. If $\mathcal{A}_2$ is marked defeated along the argumentation line $[\mathcal{A}_1, \mathcal{A}_2]$ in the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$, then $\mathcal{A}_2$ is marked defeated in the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_2)$.*

Proof. Let $\mathcal{A}_2$ be a defeater for $\mathcal{A}_1$ at $\mathcal{A}_1$ and assume $\mathcal{A}_2$ is defeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ and that $\mathcal{A}_2$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_2)$. (This is depicted in the left part of Figure 1.7. The construction given next is illustrated by the right part of the figure.) Clearly $[\mathcal{A}_1, \mathcal{A}_2]$ is an argumentation line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$, hence there is $\mathcal{A}_3$ such that $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3]$ is an argumentation line $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ with $\mathcal{A}_3$ marked undefeated.

Now, since $\mathcal{A}_2$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_2)$, $[\mathcal{A}_2, \mathcal{A}_3]$ is an argumentation line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_2)$ and $\mathcal{A}_3$ is marked defeated. Therefore, there is $\mathcal{A}_4$ such that $[\mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4]$ is an argumentation line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_2)$ with A_4 marked undefeated.

It is easy to check that if some condition (i)-(iii) from Def. 1.3.18 fails at the sequence $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4]$, then the same condition already fails either at $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3]$ or at $[\mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4]$, contradicting that these two are argumentation line. Thus we have that $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4]$ is an argumentation line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$, with $\mathcal{A}_4$ necessarily marked defeated, and hence there must exist $\mathcal{A}_5$ such that $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5]$ is an argumentation line in the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ with $\mathcal{A}_5$ necessarily marked undefeated.

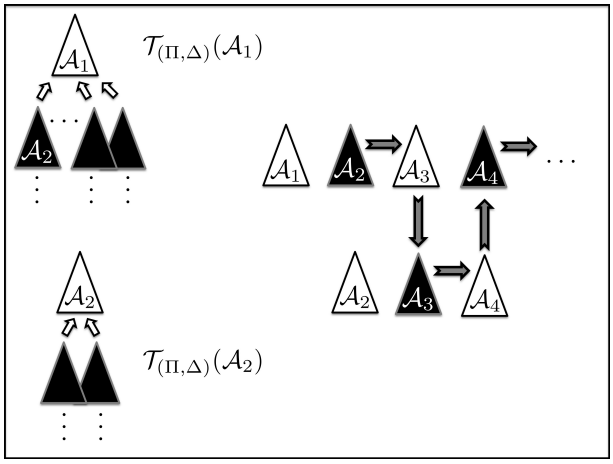


Figure 1.7: Constructing argumentation lines of arbitrary finite length: white and black triangles represent, respectively, undefeated and defeated arguments in some position. Grey arguments can be either.

Iterating this process, one can construct argumentation lines of any finite length $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots, \mathcal{A}_n, \mathcal{A}_{n+1}]$ in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$, in contradiction with Lemma 1.3.21 (2). □

Theorem 1.4.8 (Direct Cons. for Simple Programs)**.** *Given a simple program* (Π, Δ)*, the set of literals* $\mathsf{warr}(\Pi, \Delta)$ *is consistent: this set contains no pair of the form* $\langle p, t\rangle, \langle \sim p, t\rangle$*.*

Proof. Let $\langle \ell, t\rangle \in \mathsf{warr}(\Pi, \Delta)$. Thus, some argument $\mathcal{A}$ for $\langle \ell, t\rangle$ in (Π, Δ) exists that is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. Let then $\mathcal{B}$ be an arbitrary argument for $\langle \sim\ell, t\rangle$ in (Π, Δ). Assume, towards a contradiction, that $\mathcal{B}$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$.

(Case: $\mathcal{A}$ *is a proper defeater for* $\mathcal{B}$) Consider the argumentation line $[\mathcal{B}, \mathcal{A}] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$. Since $\mathcal{A}$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$, by Lemma 1.4.7, we have $\mathcal{A}$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$. Hence $\mathcal{B}$ is marked defeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$. Since $\mathcal{B}$ was arbitrary, $\langle \sim\ell, t\rangle \notin \mathsf{warr}(\Pi, \Delta)$.

(Case: $\mathcal{A}$ *is not a proper defeater for* $\mathcal{B}$). By Proposition 1.4.5 (ii), $\mathcal{B}$ is a defeater for $\mathcal{A}$, so $[\mathcal{A}, \mathcal{B}]$ is an arg. line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. Since $\mathcal{A}$ is undefeated in this tree, $\mathcal{B}$ must be defeated in this tree. Then again by Lemma 1.4.7, we have $\mathcal{B}$ is also defeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$. Since $\mathcal{B}$ was arbitrary, $\langle \sim\ell, t\rangle \notin \mathsf{warr}(\Pi, \Delta)$.

□

Similar results to those of Lemma 1.4.7 and Theorem 1.4.8 are made for the case of DeLP in [136, Props. 1 and 2]. Also, the later results in Section 1.5 are in line with [35] for defeasible logics. However, t-DeLP does not seem to reduce to the logical frameworks considered in these two contributions.

In Lemma 1.4.7, we showed the property of *being defeated* for $\mathcal{A}_2$ is downward preserved from $[\mathcal{A}_1, \mathcal{A}_2]$ to $[\mathcal{A}_2]$. (Or, conversely, *being undefeated* is upward preserved from $[\mathcal{A}_2]$ to any existing line of the form $[\mathcal{A}_1, \mathcal{A}_2]$.) This downward property can be generalized to *defeated* arguments in arbitrary interfering positions. Another upward preservation result holds for *undefeated* arguments in supporting positions.

Corollary 1.4.9. *Let* $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}]$ *be an argumentation line in* $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. *Then*

(1) if $\mathcal{A} = \mathcal{A}_{2n+1}$ *is undefeated in* Λ, *then in the corresponding arg. line* $[\mathcal{A}_2, \ldots, \mathcal{A}]$ *the (now interfering) argument* $\mathcal{A}$ *is undefeated;*

(2) if $\mathcal{A} = \mathcal{A}_{2n}$ *is defeated in* Λ, *then in the corresponding arg. line* $[\mathcal{A}_2, \ldots, \mathcal{A}]$ *the (now supporting)* $\mathcal{A}$ *is defeated.*

Finally, we address the postulate of Sub-arguments for simple t-DeLP programs. The next Lemma, though, also holds for t-DeLP programs in general.

Lemma 1.4.10. *Let* (Π, Δ) *be a t-DeLP program, and* $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ *an argumentation line in* $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. *If* $\mathcal{A}'_1$ *is an argument satisfying*

$$\mathcal{A}_1 \supseteq \mathcal{A}'_1 \supseteq \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$$

then $\Lambda' = [\mathcal{A}'_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ *is an argumentation line in* $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}'_1)$.

Conversely, if such an argumentation line Λ' *is in* $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}'_1)$ *and* $\mathcal{A}_1 \supseteq \mathcal{A}'_1$ *is an argument such that* $\Pi \cup \mathcal{A}_1 \cup \bigcup_{1 \leq i} \mathcal{A}_{2i+1}$ *is consistent, then* $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots]$ *is in* $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}'_1)$.

Proof. For the first claim, we check that Λ' satisfies conditions (i)-(iii) from Definition 1.3.18. Condition (i) holds for supporting arguments, since if $\mathcal{A}'_1 \cup \mathcal{A}_3 \cup \ldots \cup \Pi$ is inconsistent so will be $\mathcal{A}_1 \cup \mathcal{A}_3 \cup \ldots \cup \Pi$, provided that $\mathcal{A}'_1 \subseteq \mathcal{A}_1$; the

same condition trivially holds for interfering arguments, since these are exactly the same between the two argumentation lines Λ and Λ'.

For condition (ii), we first consider $\mathcal{A}_1$ and $\mathcal{A}'_1$. Let $\mathcal{A}_{2k+1}$ be an arbitrary defending argument. Note that by assumption, we have $\mathcal{A}_{2k+1} \neq \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$, which combined with the fact $\mathcal{A}'_1(\sim\mathsf{concl}(\mathcal{A}_2)) = \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$ implies $\mathcal{A}_{2k+1} \neq \mathcal{A}'_1(\sim\mathsf{concl}(\mathcal{A}_2))$. For each remaining defending argument $\mathcal{A}_3, \ldots$ and the subsequent defending arguments $\mathcal{A}_{2i+1}$, condition (ii) is obviously preserved since these arguments are exactly the same between Λ and Λ'. The same can be said about each interfering argument $\mathcal{A}_2, \ldots$ and its subsequent interfering arguments $\mathcal{A}_{2i+2}$.

Finally, for condition (iii), the only interesting case is the triple $[\mathcal{A}'_1, \mathcal{A}_2, \mathcal{A}_3]$ since the remaining triples $[\ldots, \mathcal{A}_n, \mathcal{A}_{n+1}, \mathcal{A}_{n+2}, \ldots]$ are the same between the two argumentation lines. Now, note that the properties of being a proper or blocking defeater only depend on the attacked sub-arguments, which for the triple $[\mathcal{A}'_1, \mathcal{A}_2, \mathcal{A}_3]$, are $\mathcal{A}'_1(\sim\mathsf{concl}(\mathcal{A}_2))$ and $\mathcal{A}_2(\sim\mathsf{concl}(\mathcal{A}_3))$. But these two sub-arguments are the same between the two argumentation lines: this is obvious for the latter $[\cdot, \mathcal{A}_2, \mathcal{A}_3]$; and for the former pair $[\mathcal{A}_1, \mathcal{A}_2, \cdot]$ because we have that $\mathcal{A}'_1(\sim\mathsf{concl}(\mathcal{A}_2)) = \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$, given the assumption $\mathcal{A}'_1 \supseteq \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$, so we are done.

The proof of the converse claim for Λ is analogous, except for condition (i), which is obvious given the extra assumption that $\mathcal{A}_1 \cup \mathcal{A}_3 \cup \ldots \cup \Pi$ is consistent. □

Corollary 1.4.11 (Sub-arguments for Simple Programs)**.** *Given a t-DeLP program (Π, Δ), if $\mathcal{A}_1$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ and $\mathcal{A}'_1 \subseteq \mathcal{A}_1$ is an argument, then $\mathcal{A}'_1$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}'_1)$.*

Proof. Let $\mathcal{A}'_1 \subseteq \mathcal{A}_1$ be an argument. It suffices to show that for each argument $\mathcal{A}_2$ such that $[\mathcal{A}'_1, \mathcal{A}_2, \ldots]$ is an argumentation line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}'_1)$, this argument $\mathcal{A}_2$ is defeated. So let $\mathcal{A}_2$ be an arbitrary defeater for $\mathcal{A}'_1$, i.e. let $[\mathcal{A}'_1, \mathcal{A}_2]$ be arbitrary in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}'_1)$. We show that such $\mathcal{A}_2$ is defeated.

(Case $\mathcal{A}'_1 \supseteq \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$.) Assume, towards a contradiction, that $\mathcal{A}_2$ is undefeated in $\Lambda'_2 = [\mathcal{A}'_1, \mathcal{A}_2]$. As in Lemma 1.4.7, the proof consists in the construction of argumentation lines of arbitrary finite length. Since being a defeater only depends on the defeater and the sub-argument attacked by it, the case assumption implies that $\Lambda_2 = [\mathcal{A}_1, \mathcal{A}_2]$ is also an argumentation line. Since $\mathcal{A}_1$ is undefeated, $\mathcal{A}_2$ must be defeated so there exists an argumentation line $\Lambda_3 = [\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3]$ evaluated as $[U, D, U]$. We check that $\Lambda'_3 = [\mathcal{A}'_1, \mathcal{A}_2, \mathcal{A}_3]$ is an argumentation line. Condition (i) is obvious from the fact that $\mathcal{A}_1 \supseteq \mathcal{A}'_1$. Condition (ii) is preserved from Λ_3 to Λ'_3, since the case assumption and condition (ii) for Λ_3 jointly imply

$$\mathcal{A}'_1(\sim\mathsf{concl}(\mathcal{A}_2)) = \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2)) \neq \mathcal{A}_3$$

Finally, condition (iii) for Λ'_3 also follows from the identity $\mathcal{A}'_1(\sim\mathsf{concl}(\mathcal{A}_2)) = \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$ and the fact that Λ_3 satisfies (iii). Thus, $\Lambda'_3 = [\mathcal{A}'_1, \mathcal{A}_2, \mathcal{A}_3]$ is in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}'_1)$. Since $\mathcal{A}_2$ is undefeated in Λ'_3, we have $\mathcal{A}_3$ must be defeaeted, so

an argumentation line $\Lambda'_4 = [\mathcal{A}'_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4]$ exists with $\mathcal{A}_4$ undefeated. This concludes the proof of the Base Case, where from Λ'_2 we built Λ'_4. The proof for the Inductive Case $\Lambda'_{2k} \mapsto \Lambda'_{2k+2}$ proceeds analogously and will not be repeated.

(Case $\mathcal{A}'_1 \not\supseteq \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2))$.) Consider first the Sub-Case $\sim\mathsf{concl}(\mathcal{A}_2)) \notin \mathsf{literals}(\mathcal{A}'_1)$. Then, $[\mathcal{A}'_1, \mathcal{A}_2]$ is not an argumentation line, so we are done. For the other Sub-Case, namely $\sim\mathsf{concl}(\mathcal{A}_2) \in \mathsf{literals}(\mathcal{A}'_1)$, we show that this sub-case is impossible. The present sub-case, together with the case assumption, implies that two derivations for $\sim\mathsf{concl}(\mathcal{A}_2)$ exist within $\mathcal{A}_1$: one is the sub-argument of $\mathcal{A}_1$ defeated by $\mathcal{A}_2$. The other one is the sub-argument of $\mathcal{A}'_1$ attacked by $\mathcal{A}_2$. These two sub-arguments, say $\mathcal{A}, \mathcal{A}'$, must be different since one satisfies $\mathcal{A}' \subsetneq \mathcal{A}'_1$ and the other satisfies $\mathcal{A} = \mathcal{A}_1(\sim\mathsf{concl}(\mathcal{A}_2)) \not\subseteq \mathcal{A}'_1$. From the existence of $\mathcal{A}, \mathcal{A}'$ we conclude that $\mathcal{A}_1$ is not an argument: indeed, $\mathcal{A}_1$ cannot satisfy the $\subseteq$-minimality conditions (3)-(4) from Def. 1.3.5; the reason for this is that $\mathcal{A} \neq \mathcal{A}'$ implies that either $\mathcal{A} \setminus \mathcal{A}'$ is non-empty or $\mathcal{A}' \setminus \mathcal{A}$ is non-empty. Suppose the former is the case. Then, $(\mathcal{A}_1 \setminus \mathcal{A}') \cup \mathcal{A}$ is a proper subset of $\mathcal{A}_1$ and moreover it satisfies the properties (1)-(2) from Def. 1.3.5. Hence, $\mathcal{A}_1$ is not an argument (contradiction). Otherwise, suppose that $\mathcal{A}' \setminus \mathcal{A}$ is non-empty. Now, $(\mathcal{A}'_1 \setminus \mathcal{A}) \cup \mathcal{A}'$ is a proper subset of $\mathcal{A}'_1$ satisfying (1)-(2), again contradicting the assumption that $\mathcal{A}'_1$ is an argument. □

Given the two Postulates shown above for simple programs in t-DeLP, the remaining Rationality Postulates (Indirect Consistency, Closure) are trivial consequences, since the set of strict rules in simple programs is empty.

Corollary 1.4.12 (Rationality Postulates for Simple Programs)**.** *The rationality postulates hold in t-DeLP for simple programs.*

1.5 A study of t-DeLP for mutex programs

The results from the previous section showed the rationality postulates for simple programs. The results for Indirect Consistency and Closure are not only non-trivial in the general case; they do not seem to hold for arbitrary t-DeLP programs (that is, programs with arbitrary strict rules). Fortunately, as noted in Section 1.2, many interesting examples can be captured using only a subclass of such strict rules, namely those induced by a family of mutex constraints $\mathbf{M} = \{X, \ldots\}$.

Recall from Section 1.2 that certain logical or conceptual constraints can be represented by such a family $\mathbf{M}$ of mutex sets. A mutex set X is a set of positive literals $X = \{p, q, r, \ldots\} \subseteq \mathsf{Var}$ expressing that these literals are pairwise incompatible. Hence each mutex set X induces a set of non-durative strict rules

$$X \qquad \longmapsto \qquad \Pi_X = \{\langle\sim p_i, t\rangle \leftarrow \langle p_j, t\rangle \mid p_i, p_j \in X\}$$

Finally, given a family of mutex sets $\mathbf{M}$ we denote by $\Pi_{\mathbf{M}}$ the union of the sets of strict rules Π_X for each $X \in \mathbf{M}$.

Let us observe that mutex sets, as presently defined, consisting of rules of the form $\langle\sim q, t\rangle \leftarrow \langle p, t\rangle$, require that the body contains a positive literal p and the

head a negative literal $\sim q$, for some pair $p, q \in X \in \mathbf{M}$. These restrictions on the negation in the body and the head of a mutex rule originate in the definition of a mutex set X as a set of variables. Generalizing this definition to a set of literals would give mutex sets of the form $X = \{p, \sim q, r, \ldots\}$; the latter mutex sets, at its turn, might induce rules of the form $\langle q, t\rangle \leftarrow \langle p, t\rangle$ or any other combination of positive or negative literals at the body and head of the induced rules. We adopt the former, simpler definition for the sake of simplicity. Indeed, one can always normalize a set of literals $X = \{p, \sim q, r, \ldots\}$ into a set of variables $\{p, q', r, \ldots\}$ in a new language.

In this section we show that the rationality postulates still hold for t-DeLP programs whose set of strict rules Π_{r} are precisely induced by some such mutex family $\mathbf{M}$.

Definition 1.5.1 (Mutex Program)**.** A *mutex program* (Π, Δ) with $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathsf{r}}$ is a t-DeLP program whose set of strict rules Π_{r} is induced by some family $\mathbf{M}$ of mutex sets:

$$\Pi_{\mathsf{r}} = \Pi_{\mathbf{M}} = \bigcup_{X \in \mathbf{M}} \Pi_X$$

In order to distinguish mutex rules from persistence rules δ_ℓ in the set Δ_{p}, we introduce the following notation. A mutex rule in Π_{r} of the form $\langle \sim q, t\rangle \leftarrow \langle p, t\rangle$ will be denoted $\delta_n^{\mathbf{M}}$ (for some numeric subindex) or also $\delta_p^{\mathbf{M}}$.

For the present case of mutex programs, the definition of defeat instantiates as follows.

Definition 1.5.2 (Defeat for mutex programs)**.** Let (Π, Δ) be a mutex program, with $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathbf{M}}$ for some mutex family $\mathbf{M}$. Let $\mathcal{A}_0, \mathcal{A}_1$ be two arguments in (Π, Δ) such that $\mathcal{A}_1$ attacks $\mathcal{A}_0$ at $\mathcal{B}$; that is, with $\mathsf{concl}(\mathcal{B}) = \langle \ell, t\rangle$ and $\mathsf{concl}(\mathcal{A}_1) = \langle \sim \ell, t\rangle$. We say $\mathcal{A}_1$ is a *proper defeater* for $\mathcal{A}_0$ iff

- $\mathsf{base}(\mathcal{A}_1) \supsetneq \mathsf{base}(\mathcal{B})$, or
- $\mathcal{A}_1 \cap \mathcal{B}$ is an argument for some $\langle \ell^*, t'\rangle$ with $t' < t$ and $\mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$.

Blocking defeat, denoted $\mathcal{A}_0 \prec\!\!\succ \mathcal{A}_1$, is defined by the conditions $\mathcal{A}_1 \not\succ \mathcal{A}_0$ and $\mathcal{A}_0 \not\succ \mathcal{A}_1$.

Lemma 1.5.3. *Let $\mathcal{A}, \mathcal{B}$ be two arguments in some mutex program (Π, Δ). Let $\mathcal{A}, \mathcal{B}$ be arguments attacking each other (resp. with conclusions $\langle \ell, t\rangle$ and $\langle \sim \ell, t\rangle$), and such that $\mathcal{A} \cap \mathcal{B}$ is an argument and $\mathcal{B} \setminus \mathcal{A} \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$. Then,*

(1) $\mathcal{A} \setminus \mathcal{B} \not\subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$

(2) $\mathsf{base}(\mathcal{A}) = \mathsf{base}(\mathcal{B})$

Lemma 1.5.4. *Definitions 1.3.15 and 1.5.2 are equivalent for mutex programs.*

Again, from Proposition 1.3.16 and Lemma 1.5.4, it is immediate that the defeat relation for mutex programs is well-defined.

Corollary 1.5.5. *The following hold for any mutex program (Π, Δ).*

(1) If $\mathcal{A}_1$ is a proper defeater for $\mathcal{A}_0$ at $\mathcal{B}$, then $\mathcal{B}$ is not a defeater for $\mathcal{A}_1$.

(2) If $\mathcal{A}, \mathcal{B}$ attack each other and $\mathcal{B}$ is not a proper defeater for $\mathcal{A}$, then $\mathcal{A}$ is a defeater for $\mathcal{B}$.

Example 1.5.6. Let (Π, Δ) be a mutex program with $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathbf{M}}$, defined next. Its defeasible rules essentially describe (in rule δ) that a fact $\sim p$ fades away after 100 time units of being true. The literals p and q are mutex.

$$\begin{array}{rcll} \Pi_{\mathsf{f}} & = & \{\langle q, 0\rangle\} & \\ \Delta & = & \left\{\begin{array}{c}\langle p, t+100\rangle \prec \langle \sim p, t\rangle, \\ \langle q, t+1\rangle \prec \langle q, t\rangle\end{array}\right\} & \begin{array}{l}= \delta(t) \\ = \delta_q(t)\end{array} \\ \mathbf{M} & = & \{\, \{p, q\}\, \} & \\ \Pi_{\mathbf{M}} & = & \left\{\begin{array}{l}\langle \sim p, t\rangle \leftarrow \langle q, t\rangle, \\ \langle \sim q, t\rangle \leftarrow \langle p, t\rangle\end{array}\right\} & \begin{array}{l}= \delta_p^{\mathbf{M}}(t) \\ = \delta_q^{\mathbf{M}}(t)\end{array} \end{array}$$

Then consider the next arguments in $(\Pi_{\mathsf{f}} \cup \Pi_{\mathbf{M}}, \Delta)$

$\mathcal{A} = \{\langle q, 0\rangle, \delta_q^{\mathbf{M}}(0), \delta(0)\}$ $\mathsf{concl}(\mathcal{A}) = \langle p, 100\rangle$	$\mathcal{A}^+ = \mathcal{A} \cup \{\delta_p^{\mathbf{M}}(100)\}$ $\mathsf{concl}(\mathcal{A}^+) = \langle \sim q, 100\rangle$
$\mathcal{B} = \{\langle q, 0\rangle\} \cup \{\delta_q(t)\}_{1 \le t < 100}$ $\mathsf{concl}(\mathcal{B}) = \langle q, 100\rangle$	$\mathcal{B}^+ = \mathcal{B} \cup \{\delta_q^{\mathbf{M}}(100)\}$ $\mathsf{concl}(\mathcal{B}^+) = \langle \sim p, 100\rangle$

The new Definition 1.5.2 allows for the intuitive result that the literals $\langle p, 100\rangle$ and $\langle \sim q, 100\rangle$ are warranted. This results from the dialectical trees using Definition 1.5.2 for these arguments being:

$$\begin{array}{rcl} \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{A}) & = & \{[\mathcal{A}]\} \\ \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{A}^+) & = & \{[\mathcal{A}^+]\} \\ \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{B}) & = & \{[\mathcal{B}, \mathcal{A}^+]\} \\ \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{B}^+) & = & \left\{\begin{array}{c}[\mathcal{B}^+, \mathcal{A}^+], \\ [\mathcal{B}^+, \mathcal{A}]\end{array}\right\} \end{array}$$

In contrast, if we use for example the definition of defeat for simple programs Def. 1.4.3, this would result in the following dialectical trees

$$\begin{array}{rcl} \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{A}) & = & \{[\mathcal{A}, \mathcal{B}^+, \mathcal{A}^+]\} \\ \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{A}^+) & = & \{[\mathcal{A}^+, \mathcal{B}^+]\} \\ \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{B}) & = & \{[\mathcal{B}, \mathcal{A}^+]\} \\ \mathcal{T}_{(\Pi \cup \Pi_{\mathbf{M}}, \Delta)}(\mathcal{B}^+) & = & \{[\mathcal{B}^+, \mathcal{A}^+]\} \end{array}$$

In summary, using Definition 1.4.3 would only allow us to conclude that $\langle p, 100\rangle \in \mathsf{warr}(\Pi_{\mathsf{f}} \cup \Pi_{\mathbf{M}}, \Delta)$. The literal $\langle \sim q, 100\rangle$, which follows from the latter using a strict rule would not be warranted.

Lemma 1.5.7. *Let $\mathcal{A}, \mathcal{B}$ be arguments in some mutex program (Π, Δ), with $\{\mathsf{concl}(\mathcal{A}), \mathsf{concl}(\mathcal{B})\} \subseteq X \in \mathbf{M}$. Define*

$$\begin{aligned} \mathcal{A}^+ &= \mathcal{A} \cup \{\sim\mathsf{concl}(\mathcal{B}) \leftarrow \mathsf{concl}(\mathcal{A})\} \\ \mathcal{B} &= \mathcal{B} \cup \{\sim\mathsf{concl}(\mathcal{A}) \leftarrow \mathsf{concl}(\mathcal{B})\} \end{aligned}$$

Assume that $\mathcal{A}^+$ and $\mathcal{B}^+$ are arguments in (Π, Δ). Then the following equivalences hold:

$$\mathcal{A}^+ \succ \mathcal{B} \quad \textit{iff} \quad \mathcal{A} \succ \mathcal{B}^+ \qquad \textit{and} \qquad \mathcal{A}^+ \prec\succ \mathcal{B} \quad \textit{iff} \quad \mathcal{A} \prec\succ \mathcal{B}^+$$

Theorem 1.5.8 (Rationality Postulates for Mutex Programs)**.** *Let (Π, Δ) be a mutex program with $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathbf{M}}$, for some mutex family $\mathbf{M}$. Then $\mathsf{warr}(\Pi, \Delta)$ satisfies the Rationality Postulates.*

Proof. The proof for Direct Consistency is exactly the same than for Theorem 1.4.8 (except that we use Corollary 1.5.5 instead of Corollary 1.4.5). The proof for sub-arguments is the same than in Corollary 1.4.11. It only remains to be shown the postulates of Indirect Consistency and Closure.

(Indirect Consistency) The proof is similar to Lemma 1.4.7 since we assume the contrary of the postulate and show how to build argumentation lines of arbitrary finite length. Assume then, towards a contradiction, that $\mathrm{Cn}(\mathsf{warr}(\Pi, \Delta) \cup \Pi)$ is inconsistent, so this set contains a pair, say, $\langle p, t\rangle$ and $\langle \sim p, t\rangle$. Because of Direct Consistency, one of these two literals is not in $\mathsf{warr}(\Pi, \Delta)$. Since the only rules in Π are those in the set $\Pi_{\mathbf{M}}$, and all these rules $\delta \in \Pi_{\mathbf{M}}$ have a positive literal in $\mathsf{body}(\delta)$ and a negative literal in $\mathsf{head}(\delta)$, no pair rules δ, δ' in $\Pi_{\mathbf{M}}$ can be chained: $\mathsf{head}(\delta) \notin \mathsf{body}(\delta')$. Thus, the previous conflicting literal $\langle \sim p, t\rangle$ must be derived using a (single) rule in $\Pi_{\mathbf{M}}$ and using a single "strict fact" from $\mathsf{warr}(\Pi, \Delta)$, say $\langle q, t\rangle$, for some $\{p, q\} \subseteq X \in \mathbf{M}$. Using Direct Consistency, the other conflicting literal $\langle p, t\rangle$ is the one in $\mathsf{warr}(\Pi, \Delta)$.

Existence of $\mathcal{A}$. From $\langle p, t\rangle \in \mathsf{warr}(\Pi, \Delta)$ it can be inferred that an argument $\mathcal{A}$ for $\langle p, t\rangle$ exists in (Π, Δ), and moreover, that it is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$.

Construction of $\mathcal{A}^+$. This undefeated argument $\mathcal{A}$ can be expanded into an argument $\mathcal{A}^+$ for $\langle \sim q, t\rangle$ simply by adding the rule $\langle \sim q, t\rangle \leftarrow \langle p, t\rangle \in \Pi_{\mathbf{M}}$. To see that it is indeed an argument we check the conditions (1)-(4) of Def. 1.3.5.

(1) the derivability of $\langle \sim q, t\rangle$ is obvious.

(2) assume the contrary, so the closure of $\mathcal{A}^+ \cup \Pi$ is inconsistent; then, a strict argument for some literal $\langle \ell, t'\rangle$ exists, whose negation $\langle \sim\ell, t'\rangle$ is in $\mathsf{literals}(\mathcal{A}^+)$; this literal $\langle \sim\ell, t'\rangle$ cannot be in $\mathcal{A}$, since this is an argument, so it only remains the possibility that $\langle \sim\ell, t'\rangle = \langle \sim q, t\rangle$, and so a strict argument $\Pi' \subseteq \Pi$ for $\langle q, t\rangle$ exists. But this can be expanded into a strict argument $\Pi'' = \Pi' \cup \{\langle \sim p, t\rangle \leftarrow \langle q, t\rangle\}$ for $\langle \sim p, t\rangle$, thus contradicting the fact that $\mathcal{A}$ is an argument.

(3) the $\subseteq$-minimality of $\mathcal{A}^+ \cap \Delta$ derives from the $\subseteq$-minimality of $\mathcal{A} \cap \Delta$ plus the fact that the new rule in $\mathcal{A}^+$ is strict

(4) the $\subseteq$-minimality of $\mathcal{A}^+ \cap \Pi$ also derives from that of $\mathcal{A}$.

Existence of $\mathcal{B}$. On the other hand, $\langle \sim p, t\rangle \in \mathrm{Cn}(\mathsf{warr}(\Pi, \Delta) \cup \Pi)$ implies the existence of a derivation Γ using premises from $\mathsf{warr}(\Pi, \Delta)$ and rules from $\Pi_{\mathbf{M}}$. As we mentioned above, a $\subseteq$-minimal such derivation can only be of the form $\Gamma = \{\langle q, t\rangle\} \cup \{\langle \sim p, t\rangle \leftarrow \langle q, t\rangle\}$ for some literal $\langle q, t\rangle \in \mathsf{warr}(\Pi, \Delta)$. Hence, we can use the fact that $\langle q, t\rangle \in \mathsf{warr}(\Pi, \Delta)$ to conclude the existence an argument $\mathcal{B}$ for $\langle q, t\rangle$ which is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$.

Construction of $\mathcal{B}^+$. The previous derivation for $\langle \sim p, t\rangle$ plus the argument $\mathcal{B}$ can also be merged into a derivation, say $\mathcal{B}^+ = \mathcal{B} \cup \{\langle \sim q, t\rangle \leftarrow \langle p, t\rangle\}$ for $\langle \sim p, t\rangle$. This derivation $\mathcal{B}^+$ is again an argument (the proof for this is analogous to the proof above that $\mathcal{A}^+$ is an argument).

In summary, there exist two arguments $\mathcal{A}, \mathcal{B}$ undefeated in the respective dialectical trees $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}), \mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$; and an arguments $\mathcal{A}^+$ attacking $\mathcal{B}$, and finally an argument $\mathcal{B}^+$ attacking $\mathcal{A}$.

Now, the proof by induction proceeds by cases. We only show the Base Case. The rest of the proof just follows the same steps. From here on, conditions (i)-(iii) will refer to those conditions from Def. 1.3.18.

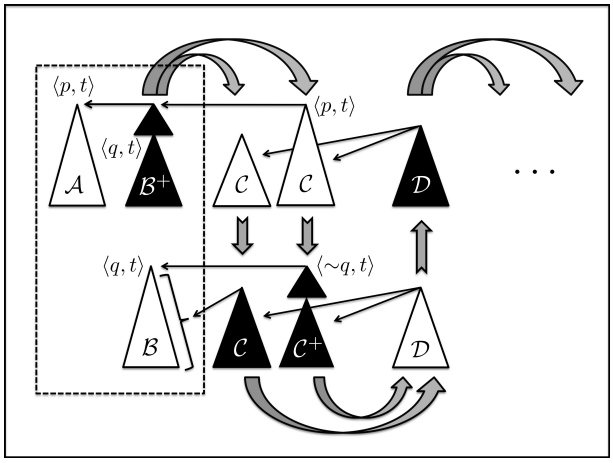


Figure 1.8: An illustration of the proof for Indirect Consistency: case $\mathcal{B}^+$ defeats $\mathcal{A}$; i.e. $\mathcal{B}^+ \succ \mathcal{A}$ or $\mathcal{B}^+ \prec\succ \mathcal{A}$.

(Case $\mathcal{B}^+ \succ \mathcal{A}$ or $\mathcal{B}^+ \prec\succ \mathcal{A}$) See Figure 1.8 for an illustration of the construction shown next. Initially we define the argumentation line $\Lambda_1 = [\mathcal{A}]$ in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$; by the case assumption, $\Lambda_2 = [\mathcal{A}, \mathcal{B}^+]$ is obviously an argumentation line. But since $\mathcal{A}$ is undefeated, an undefeated argument $\mathcal{C}$ exists with $\Lambda_3 = [\mathcal{A}, \mathcal{B}^+, \mathcal{C}]$ in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. We proceed to show that this $\mathcal{C}$ is or induces a defeater for $\mathcal{B}$, i.e. some argumentation line $[\mathcal{B}, \cdot]$ in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$. (Sub-Case $\mathcal{C}$ defeats $\mathcal{B}^+$ at $\mathcal{B}^+$) Then $\mathsf{concl}(\mathcal{C}) = \langle p, t\rangle$, so defining $\mathcal{C}^+ = \mathcal{C} \cup \{\langle \sim q, t\rangle \leftarrow \langle p, t\rangle\}$ is an argument (shown as above for $\mathcal{A}^+$). Moreover, by Lemma 1.5.7 $\mathcal{C}^+$ defeats $\mathcal{B}$ so, $[\mathcal{B}, \mathcal{C}^+]$ is in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$. We will rename such $\mathcal{C}^+$ again as $\mathcal{C}$. (Sub-Case

$\mathcal{C}$ defeats $\mathcal{B}^+$ at some sub-argument of $\mathcal{B}$) By Lemma 1.3.21 (1), $[\mathcal{B}^+, \mathcal{C}]$ is in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B}^+)$ and then by Lemma 1.4.10 $[\mathcal{B}, \mathcal{C}]$ is in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{B})$.

In either Sub-Case, we found an argumentation line $[\mathcal{B}, \mathcal{C}]$. But since this $\mathcal{B}$ is undefeated, some argumentation line of the form $[\mathcal{B}, \mathcal{C}, \mathcal{D}]$ must exist. We check that $[\mathcal{B}^+, \mathcal{C}, \mathcal{D}]$ is also an argumentation line, i.e. conditions (i)-(iii) hold. For condition (i), assume otherwise that $\mathcal{B}^+ \cup \mathcal{D} \cup \Pi$ is inconsistent. Since $\mathcal{B} \cup \mathcal{D} \cup \Pi$ is consistent, the previous inconsistency can only be with $\langle \sim p, t\rangle = \mathsf{concl}(\mathcal{B}^+)$. So $\langle p, t\rangle \in \mathrm{Cn}(\mathcal{B} \cup \mathcal{D} \cup \Pi)$. Moreover, since $\mathcal{B}^+$ is an argument, we know that $\langle p, t\rangle \in \mathrm{Cn}(\mathcal{B} \cup \mathcal{D} \cup \Pi_r)$. Finally, since $\langle p, t\rangle$ is a positive literal (and $\mathcal{B}, \mathcal{D}$ are already arguments), we must have $\langle p, t\rangle \in \mathrm{Cn}(\mathcal{B} \cup \mathcal{D})$; and since $\mathcal{B}^+$ is an argument for $\langle \sim p, t\rangle$, it must happen that $\langle p, t\rangle \in \mathrm{Cn}(\mathcal{D})$, so $\langle p, t\rangle \in \mathsf{literals}(\mathcal{D})$. But now, recall that $[\mathcal{B}, \mathcal{C}, \mathcal{D}]$ is an argumentation line so $\mathcal{B} \cup \mathcal{D} \cup \Pi$ is consistent. But this is impossible since $\mathsf{concl}(\mathcal{B}) = \langle q, t\rangle$, $\langle p, t\rangle \in \mathsf{literals}(\mathcal{D})$ and $\langle \sim q, t\rangle \leftarrow \langle p, t\rangle$ is a rule in Π. This shows that $[\mathcal{B}^+, \mathcal{C}, \mathcal{D}]$ satisfies condition (i). Conditions (ii) and (iii) are obvious for $[\mathcal{B}^+, \mathcal{C}, \mathcal{D}]$ since they only depend on the attacked sub-argument of $\mathcal{B}$, which is also an attacked sub-argument of $\mathcal{B}^+$.

Finally, it only remains to check that $\Lambda_4 = [\mathcal{A}, \mathcal{B}^+, \mathcal{C}, \mathcal{D}]$ is an argumentation line. For this, note that conditions (i), (ii) and (iii) are piecewise satisfied by $[\mathcal{A}, \mathcal{B}^+, \mathcal{C}]$ and $[\mathcal{B}^+, \mathcal{C}, \mathcal{D}]$. It is immediate that these facts jointly imply that conditions (i)-(iii) are satisfied by $[\mathcal{A}, \mathcal{B}^+, \mathcal{C}, \mathcal{D}]$.

Due to Lemma 1.5.5, the only remaining case to be checked is the following.

(Case $\mathcal{A} \succ \mathcal{B}^+$) For this case, it suffices to note that Lemma 1.5.7 implies that $\mathcal{A}^+ \succ \mathcal{B}$. But then the proof for this case is symmetric w.r.t. the former case, just switching the roles of $\mathcal{A}$ and $\mathcal{B}$.

For the Inductive Case, we would assume some argumentation line $\Lambda_{2k} = [\mathcal{A}, \mathcal{B}^+, \ldots, \mathcal{C}', \mathcal{D}']$ and extend it to some $\Lambda_{2k+2} = [\mathcal{A}, \mathcal{B}^+, \ldots, \mathcal{C}', \mathcal{D}', \mathcal{C}'', \mathcal{D}'']$. The proof is analogous to the previous Base Case $\Lambda_2 \mapsto \Lambda_4$ and will not be repeated.

(Closure) The proof is similar to that of Indirect Consistency, see Figure 1.9 for an illustration of the different possible cases in the proof. Towards a contradiction, assume that some literal $\langle \sim p, t\rangle$ is in $\mathrm{Cn}(\mathsf{warr}(\Pi, \Delta) \cup \Pi)$ but not in $\mathsf{warr}(\Pi, \Delta)$. As above, any derivation for $\langle \sim p, t\rangle$ can only consist essentially of an argument of the form $\{\langle q, t\rangle\} \cup \{\langle \sim p, t\rangle \leftarrow \langle q, t\rangle\}$, where the literal is in $\mathsf{warr}(\Pi, \Delta)$ and the rule is in $\Pi_{\mathbf{M}}$ for some some $\{p, q\} \subseteq X \in \mathbf{M}$. Also, let $\mathcal{A}$ be an argument for $\langle q, t\rangle$ in (Π, Δ) such that $\mathcal{A}$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. As in the proof of Ind. Consistency, $\mathcal{A}^+ = \mathcal{A} \cup \{\langle \sim p, t\rangle \leftarrow \langle q, t\rangle\}$ can be shown to be an argument in (Π, Δ). It only remains to be shown that $\mathcal{A}^+$ is undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}^+)$. For this, assume the contrary. We proceed to build argumentation lines of any finite length of the form: $[\mathcal{A}^+, \ldots]$ and $[\mathcal{A}, \ldots]$.

Since $\mathcal{A}^+$ is defeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}^+)$, we have some defeater $\mathcal{B}$ for it exists which is undefeated, that is, in the argumentation line $[\mathcal{A}^+, \mathcal{B}]$. These initial arguments are depicted in Fig. 1.9 within the dotted rectangle. We distinguish the next two cases, in order to show the existence of an argumentation line $[\mathcal{A}^+, \mathcal{B}, \mathcal{C}]$.

(Case $\mathcal{B}$ defeats $\mathcal{A}^+$ at some sub-argument of $\mathcal{A}$). This will lead to the top line

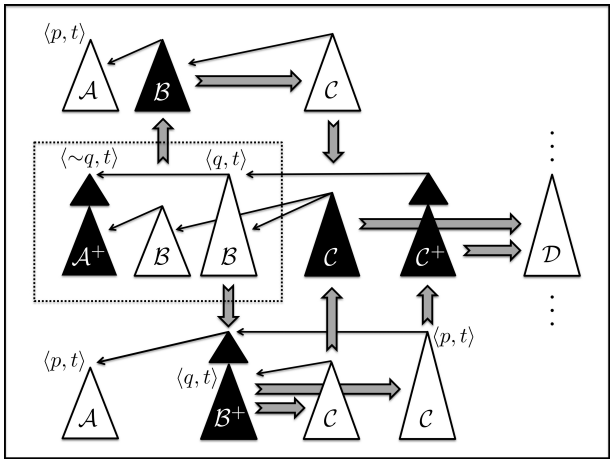


Figure 1.9: An illustration of the proof for Closure. The dotted rectangle represents the initial assumptions. The auxiliary constructions in each of the two cases considered are represented resp. by the top and bottom argumentation lines.

construction in Fig. 1.9. By the Case assumption and Lemma 1.4.10, $[\mathcal{A},\mathcal{B}]$ is also an argument, but since $\mathcal{A}$ is undefeated, $\mathcal{B}$ must be defeated, so an argument $\mathcal{C}$ exists undefeated in $[\mathcal{A},\mathcal{B},\mathcal{C}]$. We check conditions (i)-(iii) are preserved from this argumentation line to $[\mathcal{A}^+,\mathcal{B},\mathcal{C}]$. For condition (i), as in the proof of Ind. Cons., we have that the assumption that $\mathcal{A}^+\cup\mathcal{C}\cup\Pi$ is inconsistent implies that $\langle q,t\rangle \in \mathsf{literals}(\mathcal{C})$, in which case $[\mathcal{A},\mathcal{B},\mathcal{C}]$ would also violate condition (i) since $\langle\sim p,t\rangle \leftarrow \langle q,t\rangle$ is in Π. Conditions (ii) and (iii) are obviously preserved, as they only depend on the attacked sub-argument of $\mathcal{A}$ and $\mathcal{A}^+$, which is the same among the two argumentation lines.

(Case $\mathcal{B}$ defeats$\mathcal{A}^+$ at $\mathcal{A}^+$.) We denote this argument by $\mathcal{B}$, so again we have an argumentation line $[\mathcal{A}^+,\mathcal{B}]$. This case leads to the bottom line construction of Fig. 1.9. Note this argument's conclusion can only be $\mathsf{concl}(\mathcal{B}) = \langle p,t\rangle$. Thus, we can extend it into an argument $\mathcal{B}^+ = \mathcal{B}\cup\{\langle\sim p,t\rangle \leftarrow \langle q,t\rangle\}$ which attacks $\mathcal{A}$ at $\mathcal{A}$. Moreover, by Lemma 1.5.7 $[\mathcal{A},\mathcal{B}^+]$ is an argumentation line, but here $\mathcal{B}^+$ is defeated, so an argument $\mathcal{C}$ exists which is a defeater for $\mathcal{B}^+$; that is, $[\mathcal{A},\mathcal{B}^+,\mathcal{C}]$ is in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. Again we distinguish two sub-cases. (Sub-Case $\mathcal{C}$ defeats $\mathcal{B}^+$ at a sub-argument of $\mathcal{B}$.) In this case, by Lemma 1.3.21 (1), we have that $[\mathcal{B}^+,\mathcal{C}]$ is an argumentation line, and by Lemma 1.4.10, so is $[\mathcal{B},\mathcal{C}]$. Now we check that $[\mathcal{A}^+,\mathcal{B},\mathcal{C}]$ is in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}^+)$. Condition (i) is shown as usual, if $\mathcal{A}^+\cup\mathcal{C}\cup\Pi$ is inconsistent, again this can only be because $\langle q,t\rangle \in \mathsf{literals}(\mathcal{C})$. But then $\mathcal{A}\cup\mathcal{C}\cup\Pi$ would already be inconsistent, contradicting that $[\mathcal{A},\mathcal{B}^+,\mathcal{C}]$ is an argumentation line. For condition (ii), assume towards a contradiction that $\mathcal{C} = \mathcal{A}^+(= \mathcal{A}^+(\sim\mathsf{concl}(\mathcal{C})))$. Then, either we have that $\mathcal{A} \prec \mathcal{B}^+$, in which case by Lemma 1.5.7 $[\mathcal{A},\mathcal{B}^+,\mathcal{C}]$ cannot be in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$ because $\mathcal{C} = \mathcal{A}^+$ is not a defeater for $\mathcal{B}^+$; or $\mathcal{A} \prec\succ \mathcal{B}^+$, and again by Lemma 1.5.7 $[\mathcal{A},\mathcal{B}^+,\mathcal{C}] = [\mathcal{A},\mathcal{B}^+,\mathcal{A}^+]$ does

not satisfy condition (iii). In either case we reach a contradiction, so condition (ii) is satisfied by $[\mathcal{A}^+, \mathcal{B}, \mathcal{C}]$. Finally, condition (iii) is preserved from $[\mathcal{A}, \mathcal{B}^+, \mathcal{C}]$ to $[\mathcal{A}^+, \mathcal{B}, \mathcal{C}]$ due to Lemma 1.5.7.

In either case, we showed how to expand the argumentation line $[\mathcal{A}^+, \mathcal{B}]$ into $[\mathcal{A}^+, \mathcal{B}, \mathcal{C}]$. Now, since $[\mathcal{A}^+, \mathcal{B}]$ is evaluated as $[U, D]$, we must have that $[\mathcal{A}^+, \mathcal{B}, \mathcal{C}]$ is evaluated as $[U, D, U]$ and hence an argumentation line $[\mathcal{A}^+, \mathcal{B}, \mathcal{C}, \mathcal{D}]$ exists.

The proof for the inductive case is analogous and will not be repeated. □

1.6 A comparison of t-DeLP with Dung semantics, DeLP and TDR

We conclude this chapter with a detailed comparison of t-DeLP and related frameworks. First we discuss the relationship between the dialectical tree based-semantics of t-DeLP and Dung acceptability semantics for abstract argumentation frameworks. Then we report on the comparison of some particular aspects of t-DeLP with DeLP [61] and also with another temporal extension of DeLP in the literature called Temporal Defeasible Reasoning (TDR) [14].

1.6.1 t-DeLP and Dung acceptability semantics

Let us briefly review the *abstract argumentation frameworks* proposed by [52] (see also Appendix C). The latter simply consisting of a relation R, called *attack*, in set of (unstructured) elements $\mathbb{A} = \{\mathcal{A}, \ldots\}$, called *arguments*, i.e. a pair $(\mathbb{A}, R)$. The so-called acceptability semantics try to capture different intuitions about which subsets $\mathbb{E} \subseteq \mathbb{A}$ are collectively acceptable (called *extensions*), given the attack relation. For example,

$$\mathbb{E} \text{ is } \textit{conflict-free} \text{ iff no } \mathcal{A}, \mathcal{B} \in \mathbb{E} \text{ exist with } R(\mathcal{A}, \mathcal{B}).$$

Other intuitive conditions upon extensions are defined from the notion of defense: a subset $\mathbb{E}$ *defends* $\mathcal{A}$ iff

$$\text{for each } \mathcal{B} \text{ attacking } \mathcal{A}, \text{ there exists } \mathcal{C} \in \mathbb{E} \text{ that attacks } \mathcal{B}.$$

These two conditions define the set of admissible extensions or admissible semantics. Further conditions have been proposed in the literature to define different semantics based on this notion of admissibility, see Section C.1. For each semantics $\mathcal{X} = \{\text{admissible}, \ldots\}$, the (skeptical) *justified conclusions* according to $\mathcal{X}$ are defined as the conclusions of arguments in the intersection of all the $\mathcal{X}$-extensions: $\bigcap\{\mathbb{E} \mid \mathbb{E} \text{ is an } \mathcal{X}-\text{extension}\}$.

If we directly rephrase the acceptability semantics from [52] and the related definitions above, there is still a mismatch between Dung's acceptability and acceptability in t-DeLP (i.e. undefeated arguments). To see this, first note that the abstract notion of attack R would correspond to our notion of defeat. But argumentation lines in t-DeLP are not simply chains of the defeat relations, since we imposed further conditions upon the former. They are *relative to* some dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. Correspondingly, the (relevant) defense of an argument

$\mathcal{A}$ will take place only in its own dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$, so the $\mathcal{F}$ function would be expressed by

$$\mathcal{F}(\mathbb{E}) = \{\mathcal{A} \mid \forall \mathcal{B} \in \mathbb{A}\ \exists \mathcal{C} \in \mathbb{E}\ ([\mathcal{A},\mathcal{B}] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})\ \Rightarrow\ [\mathcal{A},\mathcal{B},\mathcal{C}] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})\}$$

Also, note that in a t-DeLP program (Π, Δ) there is a unique notion of *extension*, or set of "acceptable" arguments; namely, those arguments $\mathcal{A}$ that are undefeated in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A})$. That the t-DeLP procedure for undefeated arguments can define a non-admissible extension $\mathbb{E} \not\subseteq \mathcal{F}(\mathbb{E})$ is shown next.

Example 1.6.1. Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be the arguments for, resp., $\langle s, 1\rangle$, $\langle \sim s, 1\rangle$ and $\langle s, 1\rangle$ consisting of: (1) a single rule each $\langle s, 1\rangle \multimap \langle p, 0\rangle$ and $\langle \sim s, 1\rangle \multimap \langle p, 0\rangle$, $\langle q, 0\rangle$ and $\langle s, 1\rangle \multimap \langle r, 0\rangle$, resp., and (2) the facts given by the body of the corresponding rule. Moreover, let (Π, Δ) be the program defined just by these strict facts and defeasible rules from (1) and (2). Then,

$$\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}) = \{[\mathcal{A},\mathcal{B},\mathcal{C}]\} \quad \mathcal{T}_{(\Pi,\Delta)}(\mathcal{B}) = \{[\mathcal{B},\mathcal{C}]\} \quad \mathcal{T}_{(\Pi,\Delta)}(\mathcal{C}) = \{[\mathcal{C},\mathcal{B}]\}$$

so $\mathcal{A}$ is undefeated and $\mathcal{B}, \mathcal{C}$ are defeated; but there is only $\mathcal{C}$ to defend $\mathcal{A}$ (from $\mathcal{B}$). As a result, $\mathbb{E} = \{\mathcal{A}\} \not\subseteq \emptyset = \mathcal{F}(\mathbb{E})$, and thus extension $\mathbb{E}$ is not admissible.

Since complete extensions are defined by strengthening the condition for admissible extensions, namely $\mathbb{E} = \mathcal{F}(\mathbb{E})$, this counter-example also shows that the t-DeLP extensions do not correspond to any of the semantics, since they are all based on complete extensions. This includes the remaining four semantics: complete, grounded, preferred and stable.

1.6.2 Defeat criteria in DeLP and t-DeLP.

Both DeLP and t-DeLP are defeasible argumentation-based logic programming frameworks. The former is defined by criteria of defeasibility expressing a preference for arguments with more direct inference steps. This captures the preference for {*penguins do not fly*} over {*penguins are birds*, *birds fly*}. In [61], the authors use the so-called *generalized specificity* to formalize this idea of defeat as described next. Briefly, the language of DeLP is a set of literals $\ell \in \mathsf{Var} \cup \sim\mathsf{Var}$ and arguments in [61] are of the form $\langle \mathcal{A}, \ell\rangle$ for some conclusion ℓ. Recall that, in this work, arguments are identified only in terms of the defeasible information they make use of, while abstracting from strict information.

Definition 1.6.2 (DeLP-specificity)**.** Let (Π, Δ) be a DeLP program, and let Π_{r} be the set of all strict rules from Π (i.e. not including facts.) Let $\mathcal{F}$ be the set of all literals that are derivable from (Π, Δ). Let $\langle \mathcal{A}_1, \ell_1\rangle$ and $\langle \mathcal{A}_2, \ell_2\rangle$ be two arguments obtained from (Π, Δ). $\langle \mathcal{A}_1, \ell_1\rangle$ is *strictly more specific* than $\langle \mathcal{A}_2, \ell_2\rangle$ if the following conditions hold:

1.	for all $H \subseteq \mathcal{F}$:	if $\Pi_{\mathsf{r}} \cup H \cup \mathcal{A}_1 \vdash \ell_1$ and $\Pi_{\mathsf{r}} \cup H \nvdash \ell_1$, then $\Pi_{\mathsf{r}} \cup H \cup \mathcal{A}_2 \vdash \ell_2$, and
2.	there exists $H' \subseteq \mathcal{F}$ such that:	$\Pi_{\mathsf{r}} \cup H' \cup \mathcal{A}_2 \vdash \ell_2$ and $\Pi_{\mathsf{r}} \cup H' \nvdash \ell_2$ and $\Pi_{\mathsf{r}} \cup H' \cup \mathcal{A}_1 \nvdash \ell_1$.

When the conditions for 1 are met, i.e. $\Pi_{\mathsf{r}} \cup H \cup \mathcal{A} \vdash \ell$ and $\Pi_{\mathsf{r}} \cup H \nvdash \ell_1$, we say H is an *activation set* for $\langle \mathcal{A}, \ell \rangle$. The idea of DeLP-specificity is to prefer arguments with fewer activation sets (in the sense of inclusion). In other words, to prefer the existence of less combinations of intermediate steps sufficing for the conclusion.

Turning back to the language of t-DeLP, if we redefine the activation sets of $\mathcal{A}$ by the set $\{\mathsf{base}(\mathcal{A})\}$, (i.e. by the unique set of strict facts $\mathcal{A} \cap \Pi_{\mathsf{f}}$), this specificity criterion in Definition 1.3.15 turns into our first criterion for t-DeLP defeat, namely the $\subseteq$-comparison between $\mathsf{base}(\mathcal{A}_1)$ and $\mathsf{base}(\mathcal{A}_2)$. On the other hand, the second criterion in Def. 1.3.15, based on the use of persistence rules, can also be expressed using activation sets $\mathit{activ}(\mathcal{A})$ of an argument $\mathcal{A}$. The difference is that now the preference is for *more activation sets*. For example, we have the following equivalence for simple programs

$$\begin{array}{c} \mathcal{A}_1 \succ \mathcal{A}_2 \\ \text{(2nd crit. of Def. 1.4.3)} \end{array} \quad \text{iff} \quad \begin{array}{c} \text{there is } \Delta'_{\mathsf{p}} \subseteq \Delta_{\mathsf{p}} \text{ s.t. that in } (\Pi, \Delta \smallsetminus \Delta'_{\mathsf{p}}) \\ \mathit{activ}(\mathcal{A}_1) \supsetneq \mathit{activ}(\mathcal{A}_2) \end{array}$$

1.6.3 t-DeLP and Temporal Defeasible Reasoning (TDR)

In the TDR framework [14], literals (hence conclusions) of arguments are primitively associated with both discrete intervals and time-points. For instance, (using our own notation) $\mathsf{head}(\delta) = \langle \ell, [t+1, t+3] \rangle$ expresses that if $\delta \in \Delta$ applies we defeasibly conclude that ℓ holds from $t+1$ to $t+3$. In TDR, conflicts between two interval-valued arguments, e.g. $(\mathcal{A}, \langle \ell, [t+1, t+4] \rangle)$ and $(\mathcal{B}, \langle \sim\ell, [t+2, t+5] \rangle)$ attacking each other do so at the intersection of these intervals $[t+2, t+4]$.

Another significant difference between TDR and t-DeLP lies again in the defeat criteria. In particular, when persistence rules are involved in the comparison between two contending arguments. In TDR, an argument that contains persistence rules is less preferred than an argument which does not. In contrast, in t-DeLP (Def. 1.3.15 above) the comparison is made in terms of set-theoretic inclusion, thus providing us with a more fine-grained comparison of persistence than the criterion proposed in TDR. Thus, in t-DeLP an argument $\mathcal{A}$ using persistence need not be properly defeated when attacked by a persistence-free argument $\mathcal{B}$. For example, if the persistent literals $\langle \ell, t \rangle, \ldots, \langle \ell, t+k \rangle$ in $\mathcal{A}$ are not directly attacked by the other argument, both arguments will be blocking defeaters for each other.

Figure 1.10 summarizes the differences between DeLP, t-DeLP and TDR. In contrast to the TDR system in (as well as other temporal argumentation systems [95] [43]), in t-DeLP we let the notion of an *interval where some conclusion holds* to be a notion deriving from the *set of time-points for which this conclusion holds*. In this sense, TDR is more expressive than t-DeLP, though for most applications, it seems possible to translate a TDR-proof for the warrant of $\langle \ell, [t, t'] \rangle$ in a given TDR program, into a t-DeLP-proof for the warrant of each $\langle \ell, t_0 \rangle$ with $t \leq t_0 \leq t'$ in a corresponding t-DeLP program.

	DeLP	t-DeLP	TDR
literals	p or $\sim p$	$\langle p, t\rangle$ or $\langle \sim p, t\rangle$	$(\neg)\text{Holds}_{at}(p, t)$ $(\neg)\text{Holds}_{in}(p, [t, t'])$
derivability	modus ponens	modus ponens	modus ponens
argument $\mathcal{A}$	$\mathcal{A} \subseteq \Delta$ $\mathcal{A} \cup \Pi \vdash \ell$	$\mathcal{A} \subseteq \Pi \cup \Delta$ $\mathcal{A} \vdash \langle \ell, t\rangle$	$\mathcal{A} \subseteq \Delta$ $\mathcal{A} \cup \Pi \vdash$ literal
$\mathcal{A}$ attacks $\mathcal{B}$	$\langle \mathcal{A}, \ell\rangle$ attacks $\langle \mathcal{B}, \ell'\rangle$ iff $\{\ell, \ell'\} \cup \Pi$ is inconsistent	$\sim\mathsf{concl}(\mathcal{A}) \in$ $\mathsf{literals}[\mathcal{B}]$	$\text{Holds}_{in}(p, I)$ vs. $\neg\text{Holds}_{in}(p, I')$, with $I \cap I' \neq \emptyset$
specificity	generalized spec. (activation sets)	Definition 1.3.15 (base = act. set, less persistence)	pointwise gen. spec. (pointwise activation sets)
more direct rules	by specificity	no	by specificity
more premises	if more specific	always	if more specific
persistence	n/a	comparison $\mathcal{A} \setminus \mathcal{B}$ vs. $\mathcal{B} \setminus \mathcal{A}$	comparison $\mathcal{A}$ vs $\mathcal{B}$
warrant	dialectical tree	dialectical tree	algorithm in [14]

Figure 1.10: A comparison of DeLP, t-DeLP and TDR.

1.7 Conclusions and Related Work

In this chapter we defined t-DeLP, a temporal version of DeLP where logic programs contain temporal literals and rules with duration. The proposed framework modifies features of DeLP in order to deal with specific issues related to temporal reasoning, like persistence and the past/future asymmetry in causal statements. Besides this differences at the definition of defeat, t-DeLP is essentially based on the same argumentation-based procedure that defines the notion of defeasible logical consequence, or warrant. This notion of the set of warranted literals of a program has been shown to satisfy the postulates of Direct Consistency and Sub-arguments (the other postulates are trivial). In addition, we have extended the basic framework to deal with programs defined by a family of mutex constraints, and we have shown the rationality postulates hold as well (but less trivially) for this class of mutex programs.

There is a vast literature on logics for reasoning about causality and time, roughly dividing into two areas: temporal logics, and causal or conditional logics. These focus respectively on the temporal aspects of change, and the causal relationships between state conditions. This is sometimes done by focusing on states and leaving events (transitions between states) without an explicit representation in the object language, as in the planning tradition.

Modal logic [40], [26] is one of the most central areas within logic in computer science, and has been used in particular for the study of time, events or actions (among many other topics). For example, studies in applied modal logic include modalities for time in linear time LTL [71], or branching time CTL, CTL* [55] [56]; or modalities for the execution of programs PDL [74].

In practice, though, the early discovery of some knowledge engineering problems motivated the study of non-monotonic reasoning. For example, the recognition of the *frame problem* can be singled out as one of the major challenges in early studies on knowledge representation. In the broad sense, this denotes a family of problems related to the description of actions: their effects, non-effects, or preconditions. Among them, we find:

- the *frame problem* -in the narrow sense- is the problem of finding (efficient) representations for the persistence of facts through time or action executions. Since an action will only change a small part of a scenario, it is unpractical to make an explicit list of which facts persist under which actions, to be used during inference.
- the *ramification problem* is the problem of efficiently deriving the indirect effects of an action in a given context. (See Section 2.2).
- the *qualification problem* is that of finding efficient representations for the preconditions of an action. We would also like to prune many of the preconditions that one would not bother to check before the action (unless one positively knows about their failure). Among the three problems in this list, only this one seems to demand a non-monotonic approach [38].

The original frame problem plagued the initial classical logic based approaches [97], etc. and it also affects many (monotonic) temporal logics, like temporal logic programming [20]. Some efforts have been devoted to solve the frame problem within monotonic modal logics. For example, some of the issues related to the frame problem in PDL have been successfully dealt with in [123], [67], [152], [38]. Other research areas, instead, have found natural ways to avoid the frame problem or address it in natural ways. Planning systems, for example, avoid the frame problem by severely restricting logical inference. Non-monotonic logics (see below). These logics have found more or less natural ways to capture common sense reasoning and avoid the former representation issues.

We hope that the examples in Chapters 1 and 2 suffice to give an idea of how these problems can be addressed in t-DeLP (with actions). In summary, the (narrow) frame problem can be solved by means of persistence rules; the ramification problem can be addressed by appropriate defeasible rules (see Section 2.2; and the qualification problem can be solved by the argumentation procedure (arguments against the precondition of an action can qualify its executability).

The present work broadly belongs to the area of non-monotonic temporal logics, where non-monotonicity here is built upon the recent area of computational argumentation [52], [121], [126] and more specifically under the form of logic programming [35], [61].

Let us then briefly survey different non-monotonic logical frameworks (see [32]), and more specifically those modeling events, action or time. Inspired by common sense reasoning, non-monotonic logics are based on the existence of priorities between inferences [122], [8], [99]. Thus, while all inferences separately make sense, some of them might be preferred to (and cancel) others. Non-monotonic logics, though, have evolved into a rather disperse variety of logical

approaches, including some modal approaches [83], [68]. Some correspondences or reductions are known among non-monotonic logical systems, among which: default logics into autoepistemic logics [83], or into DeLP [50]; DeLP into ASP [136]; the correspondence between DeLP and normal logic programing [100], defeasible logic and definite logic programming [59]. Among non-monotonic modal approaches, we find [68],[33]. Other logics of action and causation include $\mathcal{C}/\mathcal{C}+$ [69], [47], $\mathcal{A}$ [63], event calculus [88] and others. These have been studied from the standpoint of PDL in [152].

Further motivations for the present, argumentation-based approach are precisely questions on these priorities between conflicting inferences [130], [61]: how are they defined, but also how can they be automatically generated, etc. In most approaches in the literature on abstract argumentation [52], [7] this question is left unanswered. This issue becomes specially important when the internal structure of arguments is considered, e.g. in default logics [129] or in logic-based argumentation [35], [121].

The present approach was inspired by the notion of *specificity* [120], [134], and its application to DeLP [61]. Besides this work, other rule-based systems were initially proposed [25], [102] in the area of defeasible logic. These were recently extended with temporal reasoning [72]. Indeed, our language is mainly inspired by this latter work, although we opted for an argumentation-based approach. The reasons for this choice are two: first, the tools of rules and defeaters (resp., to promote and prevent derivations) is conceptually less natural than the use of arguments. than consist of rules, in a way that mirrors a deliberating human agent pondering reasons for and against candidate conclusions. Second, argumentation-based logics are more powerful than rule-based systems since priority relations can apply at a global level (the logical structure of arguments) rather than at a local level (comparisons between rules). This applies to other frameworks for temporal temporal reasoning like Hunter [80], [81].

Finally, several frameworks have been proposed in the more recent area of logical models of argumentation, following the seminal work [52]. This work has been extended by associating time intervals to arguments (applicable in these intervals) [42], [43], or at the level of rules [14], [95]. Our approach differs from these latter works in that the *interval where the conclusion of an argument holds*, rather than being a primitive notion, obtains from different arguments (one for each time-point).

The present chapter is closely related to other logical systems in the family of DeLP, namely ODeLP [36], PDeLP [3], RP-DeLP [4], TDR [14], pt-DeLP [70]. Each of this is based on different formal definitions of defeat (preference), modeling defeasible reasoning with uncertainty or time.

1.8 Appendix: proofs

The proofs for the auxiliary results mentioned in the previous sections are presented here.

Proposition. 1.3.7 Let (Π, Δ) be a t-DeLP program, and let $\mathcal{A}$ be an argument for some $\langle \ell, t\rangle = \mathsf{concl}(\mathcal{A})$. Then $\{\langle \ell, t\rangle\} = \mathsf{head}[\mathcal{A}] \setminus \bigcup \mathsf{body}[\mathcal{A}]$.

Proof. For the first claim, clearly if $\mathcal{A}$ is an argument for $\langle \ell, t\rangle$ then $\langle \ell, t\rangle$ is derivable from $\mathcal{A}$. By definition of derivability, some rule (possibly a strict fact) δ exists with $\mathsf{head}(\delta) = \langle \ell, t\rangle$. By the $\subseteq$-minimality conditions, no other rule δ' exists in $\mathcal{A}$ with $\langle \ell, t\rangle \in \mathsf{body}(\delta')$. This shows $\{\langle \ell, t\rangle\} \subseteq \mathsf{head}[\mathcal{A}] \setminus \bigcup \mathsf{body}[\mathcal{A}]$.

On the other hand, let δ'' be an arbitrary rule in $\mathcal{A}$ with $\mathsf{head}(\delta'') \notin \mathsf{body}[\mathcal{A}]$. (Case $\mathsf{head}(\delta'') = \langle \ell, t\rangle$.) Then, by $\subseteq$-minimality of $\mathcal{A}$ w.r.t. the derivability of $\langle \ell, t\rangle$ from $\mathcal{A}$, we have $\delta'' = \delta$. (Case $\mathsf{head}(\delta'') \not\leq \ell, t\rangle$.) That is, $\delta'' \neq \delta$. By definition of derivability $\mathsf{head}(\delta'') \notin \mathsf{body}[\mathcal{A}]$ implies that $\mathcal{A} \setminus \{\delta''\} \vdash \langle \ell, t\rangle$. So either we have $\delta'' \in \Delta$, in which case $\mathcal{A}_\Delta$ is not $\subseteq$-minimal with the property $\mathcal{A}_\Delta \cup \Pi \vdash \langle \ell, t\rangle$. Or, $\delta'' \in \Pi$, in which case $\mathcal{A}_\Pi$ is not $\subseteq$-minimal with $\mathcal{A}_\Delta \cup \mathcal{A}_\Pi \vdash \langle \ell, t\rangle$. In either sub-case, we reach a contradiction. This shows that $\mathsf{head}[\mathcal{A}] \setminus \bigcup \mathsf{body}[\mathcal{A}] \subseteq \{\langle \ell, t\rangle\}$. □

For the next result, we start with an inductive definition for the notion of sub-argument (Def. 1.3.9). Given an argument $\mathcal{A}$ in some t-DeLP program (Π, Δ) and a literal $\langle \ell, t\rangle \in \mathsf{literals}(\mathcal{A})$, the sub-argument of $\mathcal{A}$ for $\langle \ell, t\rangle$, denoted $\mathcal{A}(\langle \ell, t\rangle)$, is the set obtained by the following inductive construction:

if $\delta \in \mathcal{A}$ exists with $\mathsf{head}(\delta) = \langle \ell, t\rangle$,	then $\delta \in \mathcal{A}(\langle \ell, t\rangle)$
if $\delta \in \mathcal{A}(\langle \ell, t\rangle)$ and $\delta' \in \mathcal{A}$ exists with $\mathsf{head}(\delta') \in \mathsf{body}(\delta)$,	then $\delta' \in \mathcal{A}(\langle \ell, t\rangle)$

Proposition. 1.3.10 Given some argument $\mathcal{A}$ and a literal $\langle \ell, t\rangle \in \mathsf{literals}(\mathcal{A})$, then $\mathcal{A}(\langle \ell, t\rangle)$ is unique.

Proof. By induction on the complexity of $\mathcal{A}$.

(Base Case) Suppose that $\mathcal{A}$ is a strict fact $\mathcal{A} = \{\langle \ell, t\rangle\} \subseteq \Pi_{\mathsf{f}}$. Then, $\mathsf{literals}(\mathcal{A}) = \mathcal{A}$ so $\mathcal{A}(\langle \ell, t\rangle) = \mathcal{A}$ and it is the only sub-argument of $\mathcal{A}$ deriving $\langle \ell, t\rangle$. Hence it is unique.

(Ind. Case) Assume (Ind. Hyp.) that for any argument $\mathcal{A}$ with some $\delta \in \mathcal{A}$ such that $\mathsf{head}(\delta) = \mathsf{concl}(\mathcal{A})(= \langle \ell, t\rangle)$, we have $\mathcal{A}(\langle \ell', t'\rangle)$ is unique for each $\langle \ell', t'\rangle \in \mathsf{literals}(\mathcal{A})$. We check the unicity of the remaining case $\mathcal{A}(\mathsf{concl}(\mathcal{A})) = \mathcal{A}$. Suppose another sub-argument $\mathcal{B} \subseteq \mathcal{A}$ exists for $\langle \ell, t\rangle$. From $\mathcal{B} \subseteq \mathcal{A}$ and $\mathcal{B} \neq \mathcal{A}$, we infer the existence of some rule or literal $\delta' \in \mathcal{A} \setminus \mathcal{B}$. (Case $\delta' \in \Pi$) If this δ' is a literal or a strict rule, then such $\delta' \in \mathcal{A}_\Pi$ shows that $\mathcal{A}$ does not satisfy, in a $\subseteq$-minimal way, that $\mathcal{A}_\Delta \cup \mathcal{A}_\Pi \vdash \langle \ell, t\rangle$; hence $\mathcal{A}$ violates condition (4) from Def. 1.3.5, so $\mathcal{A}$ is not an argument (contradiction). (Case $\delta' \in \Delta$) Then δ' shows that $\mathcal{A}_\Delta$ does not satisfy $\Pi \cup \mathcal{A}_\Delta \vdash \langle \ell, t\rangle$ in a $\subseteq$-minimal way; so $\mathcal{A}$ does not satisfy condition (3) from Def.1.3.5. Again, $\mathcal{A}$ cannot be an argument (contradiction). □

Proposition. 1.3.16 The following hold for any t-DeLP program:

(1) If $\mathcal{A}_1$ is a proper defeater for $\mathcal{A}_0$ at $\mathcal{B}$, then $\mathcal{B}$ is not a defeater for $\mathcal{A}_1$.

(2) If $\mathcal{A}, \mathcal{B}$ attack each other, and $\mathcal{B}$ is not a proper defeater for $\mathcal{A}$, then $\mathcal{A}$ is a defeater for $\mathcal{B}$.

Proof. Claim (1). We show first that $\mathcal{A}_1$ and $\mathcal{B}$ cannot be proper defeaters for each other. First, note that we cannot have $\mathcal{A}_1 \succ \mathcal{B}$ and $\mathcal{B} \succ \mathcal{A}_1$ due to the first criterion in both cases; the reason is that otherwise one would obtain $\mathsf{base}(\mathcal{A}_1) \supsetneq \mathsf{base}(\mathcal{B}) \supsetneq \mathsf{base}(\mathcal{A}_1)$. Second, neither it can be that $\mathcal{A}_1 \succ \mathcal{B}$ and $\mathcal{B} \succ \mathcal{A}_1$ due to the second criterion. In this case, we would obtain a contradiction:

$$\begin{array}{ll} \mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi & \text{(2nd cond. from } \mathcal{A}_1 \succ \mathcal{B}) \\ \mathcal{B} \setminus \mathcal{A}_1 \cap (\Delta \setminus \Delta_{\mathsf{p}}) \neq \varnothing & \text{(4th cond. from } \mathcal{B} \succ \mathcal{A}_1) \end{array}$$

Finally, we can also rule out that, e.g. $\mathcal{A}_1 \succ \mathcal{B}$ is due to the first criterion while $\mathcal{B} \succ \mathcal{A}_1$ is due to the second (or viceversa). In this case, the former $\mathcal{A}_1 \succ \mathcal{B}$ would imply $\mathsf{base}(\mathcal{A}_1) \supsetneq \mathsf{base}(\mathcal{B})$ while the latter $\mathcal{B} \succ \mathcal{A}_1$ (using the fifth cond.) would require that either $\mathsf{base}(\mathcal{B}) \not\subseteq \mathcal{A}_1$ or $\mathsf{base}(\mathcal{B}) = \mathsf{base}(\mathcal{A})$. Either of these two cases is inconsistent with the former assumption.

This shows that the relation of proper defeat (between mutually attacking arguments $\mathcal{A}_1, \mathcal{B}$) is asymmetric. Moreover, using the definition of blocking defeat $\prec\!\!\succ$, it is obvious that if $\mathcal{A}_1$ is a proper defeater for $\mathcal{B}$, these two arguments can neither be blocking defeaters for each other.

Claim (2) is straightforward: assume that $\mathcal{B} \not\succ \mathcal{A}$. Then either $\mathcal{A} \succ \mathcal{B}$, in which case $\mathcal{A}$ is a proper defeater for $\mathcal{B}$, or $\mathcal{A} \not\succ \mathcal{B}$, in which case $\mathcal{A}$ is a blocking defeater for $\mathcal{B}$. □

Lemma. 1.3.21 For any t-DeLP program (Π, Δ),

(1) If $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \ldots, \mathcal{A}_n]$ is an argumentation line for $\mathcal{A}_1$, then $[\mathcal{A}_m, \ldots, \mathcal{A}_n]$ is an argumentation line for $\mathcal{A}_m$.

(2) Each argumentation line $\Lambda = [\mathcal{A}_1, \ldots] \in \mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ is finite. The dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ is finite.

Proof. For (1), let $\Lambda_1 = [\mathcal{A}_1, \ldots, \mathcal{A}_m, \ldots, \mathcal{A}_n]$ be an argumentation line for $\mathcal{A}_1$. Notice that the first element of Λ_m is $\mathcal{A}_m$. We check that each condition (i)-(iii) from Definition 1.3.18 holds for the sequence $\Lambda_m = [\mathcal{A}_m, \ldots, \mathcal{A}_n]$.

(i) The joint consistency of supporting (resp. interfering) arguments is satisfied by Λ_m, since otherwise if $\mathcal{A}_m \cup \ldots \cup \mathcal{A}_n \cup \Pi$ was inconsistent, then so would be $\mathcal{A}_1 \cup \ldots \cup \mathcal{A}_n \cup \Pi$, contradicting that Λ_1 is an argumentation line.

(ii) If this condition failed for Λ_m at the pair $\mathcal{A}_{m+k} = \mathcal{A}_{m+k+2j}(\sim\mathsf{concl}(\mathcal{A}_{m+k+2j+1}))$, then it would already fail for Λ_1 at the same pair.

(iii) The condition that $\mathcal{A}_{i+1}$ is a proper defeater for $\mathcal{A}_i$ if $\mathcal{A}_i$ is a blocking defeater for $\mathcal{A}_{i-1}$ must hold for Λ_k since otherwise it would also fail for Λ_1 at the same triple $\mathcal{A}_{i-1}, \mathcal{A}_i, \mathcal{A}_{i+1}$.

For (2), let $\mathsf{concl}(\mathcal{A}_1) = \langle \ell, t \rangle$. Recall that $t < \omega$ and the set Lit is also finite, so the set of literals $\langle \ell', t' \rangle$ with $t' \leq t$ is finite. In consequence, the set of rules δ whose head is some $\langle \ell', t' \rangle$ is also finite. Since arguments are finite sets of rules

and literals, the set of arguments $\mathcal{A}_{2n+1}$ whose conclusion is some $\langle \ell', t' \rangle$ with $t' \leq t$ is also finite. Hence each argumentation line for $\mathcal{A}_1$ is finite. Finally, there are a finite number of argumentation lines for $\mathcal{A}_1$ (again because the number of arguments for $t' \leq t$ are finite). The latter two facts imply that the dialectical tree $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$ is finite.

□

Lemma. 1.4.4 For any simple program (Π, Δ), Definitions 1.3.15 and 1.4.2 are equivalent.

Proof. We only need to check the equivalence between the second (persistence-based) criterion from the two definitions.

($\Rightarrow$) We have the following:

$$\begin{array}{ll} \mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi & \text{(by 2nd condition in Def. 1.3.15)} \\ \mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathsf{f}} & \text{(since } \Pi_{\mathsf{r}} = \varnothing\text{)} \\ \mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} & \text{(shown next)} \end{array}$$

To obtain the last step, assume the contrary and let $\langle \ell_0, t_0 \rangle \in \Pi_{\mathsf{f}} \cap \mathcal{B}$ be such that $\langle \ell_0, t_0 \rangle \notin \mathcal{A}_1$. Since $\mathcal{A}_1 \cap \mathcal{B}$ is an argument (the 1st condition in Def. 1.3.15), we have the next two cases. (Sub-Case: $\langle \ell_0, t_0 \rangle = \mathsf{concl}(\mathcal{A}_1 \cap \mathcal{B})$.) Then, by $\subseteq$-minimality, $\mathcal{A}_1 = \{\langle \ell_0, t_0 \rangle\}$, contradiction. (Sub-Case: $\langle \ell_0, t_0 \rangle \neq \mathsf{concl}(\mathcal{A}_1 \cap \mathcal{B})$.) Since $\Pi_{\mathsf{r}} = \varnothing$, the latter implies some rule $\delta \in \Delta_{\mathsf{p}} \cap \mathcal{B}$ exists with $\mathsf{body}(\delta) \supseteq \{\langle \ell_0, t_0 \rangle, \mathsf{concl}(\mathcal{A}_1 \cap \mathcal{B})\}$. Since these two elements are different, this contradicts the fact that $|\mathsf{body}(\delta)| = 1$, given by the assumption $\delta \in \Delta_{\mathsf{p}}$.

So we obtain $\mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}}$. Since the rules in $\mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}}$ satisfy the condition $|\mathsf{body}(\delta)| = 1$, the fact that $\mathcal{B}$ is an argument (with a unique conclusion) requires that $\mathcal{B} \setminus \mathcal{A}_1$ is of the form $\{\delta_{\ell^*}(t'')\}_{t^* \leq t'' < t}$.

On the other hand, that $\mathcal{A}_1 \cap \mathcal{B}$ is an argument implies that it is a sub-argument of $\mathcal{A}_1$, hence of the form $\mathcal{A}_1(\langle \ell^\star, t^\star \rangle)$. Using this and the claim just shown above, we reason as follows:

$$\begin{array}{lcll} \mathcal{B} & = & (\mathcal{A}_1 \cap \mathcal{B}) \cup (\mathcal{B} \setminus \mathcal{A}_1) & \text{(disjoint union)} \\ \mathcal{B} & = & \mathcal{A}_1(\langle \ell^\star, t^\star \rangle) \cup \{\delta_{\ell^*}(t'')\}_{t^* \leq t'' < t} & \text{(using the last two claims)} \\ \mathcal{B} & = & \mathcal{A}_1(\langle \ell^*, t^* \rangle) \cup \{\delta_{\ell^*}(t'')\}_{t^* \leq t'' < t} & (t^* = t^\star \text{ because this union is disjoint} \\ & & & \text{and } \mathcal{B} \text{ is } \subseteq\text{-minimal w.r.t. } \mathcal{B} \vdash \langle \ell^*, t \rangle) \end{array}$$

Finally, the 3rd condition $(\mathcal{B} \setminus \mathcal{A}_1) \cap \Delta_{\mathsf{p}} \neq \varnothing$ implies that $t^* < t$. This concludes the proof of Def. 1.4.3.

($\Leftarrow$) We first show the 4th condition from Def. 1.3.15. From the assumption $\mathcal{B} = \mathcal{A}_1(\langle \ell, t' \rangle) \cup \{\delta_\ell(t'')\}_{t' \leq t'' < t}$, we can assume without loss of generality that this t' is minimal with this property, so some $\delta^\star \in (\mathcal{A}_1 \setminus \mathcal{B}) \setminus \Delta_{\mathsf{p}}$ exists with $\langle \ell, t' \rangle \in \mathsf{body}(\delta^\star)$. Since, for simple programs, $\delta^\star \notin \Delta_{\mathsf{p}}$ implies $\delta^\star \in \Delta \setminus \Delta_{\mathsf{p}}$, this shows the 4rth condition $(\mathcal{A}_1 \setminus \mathcal{B}) \cap (\Delta \setminus \Delta_{\mathsf{p}})$. Moreover, the above minimality of t' implies that $\{\delta_\ell(t'')\}_{t' \leq t'' < t} \cap \mathcal{A}_1 = \varnothing$. Indeed, some $t' < t$ with this property

exists since $\mathsf{concl}(\mathcal{A}_1) \neq \mathsf{concl}(\mathcal{B})$. We use these latter properties of t' to reason as follows

$$\begin{array}{rcll}
\mathcal{B} & = & \mathcal{A}_1(\langle \ell, t' \rangle) \cup \{\delta_\ell(t'')\}_{t' \leq t'' < t} & \text{(Def. 1.4.3)} \\
\mathcal{A}_1 \cap \mathcal{B} & = & \mathcal{A}_1 \cap (\mathcal{A}_1(\langle \ell, t' \rangle) \cup \{\delta_\ell(t'')\}_{t' \leq t'' < t}) & \\
\mathcal{A}_1 \cap \mathcal{B} & = & (\mathcal{A}_1 \cap \mathcal{A}_1(\langle \ell, t' \rangle)) \cup (\mathcal{A}_1 \cap \{\delta_\ell(t'')\}_{t' \leq t'' < t}) & \\
\mathcal{A}_1 \cap \mathcal{B} & = & \mathcal{A}_1(\langle \ell, t' \rangle) \cup (\mathcal{A}_1 \cap \{\delta_\ell(t'')\}_{t' \leq t'' < t}) & \text{since } \mathcal{A}_1(\langle \ell, t' \rangle) \subseteq \mathcal{A}_1 \\
\mathcal{A}_1 \cap \mathcal{B} & = & \mathcal{A}_1(\langle \ell, t' \rangle) & \text{since } \mathcal{A}_1 \cap \{\delta_\ell(t'')\}_{t' \leq t'' < t} = \varnothing
\end{array}$$

Now, since $\mathcal{A}_1(\langle \ell, t' \rangle)$ is an argument for $\langle \ell, t' \rangle$ with $t' < t$, so is $\mathcal{A}_1 \cap \mathcal{B}$. This shows the 1st condition from Def. 1.3.15. The 2nd condition is obvious $\mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \subseteq \Delta_{\mathsf{p}} \cup \Pi$. The 3rd condition $(\mathcal{B} \setminus \mathcal{A}_1) \cap \Delta_{\mathsf{p}} \neq \varnothing$ follows from $t' < t$. Finally, for the 5th condition, it is straightforward that the second criterion from Def. 1.4.3 implies $\mathsf{base}(\mathcal{B}) = \mathsf{base}(\mathcal{A}_1)$ for simple programs. □

Corollary. 1.4.9 Let $\Lambda = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}]$ be an argumentation line in $\mathcal{T}_{(\Pi,\Delta)}(\mathcal{A}_1)$. Then

(1) if $\mathcal{A} = \mathcal{A}_{2n+1}$ is undefeated in Λ, then in the corresponding arg. line $[\mathcal{A}_2, \ldots, \mathcal{A}]$ the (now interfering) argument $\mathcal{A}$ is undefeated;

(2) if $\mathcal{A} = \mathcal{A}_{2n}$ is defeated in Λ, then in the corresponding arg. line $[\mathcal{A}_2, \ldots, \mathcal{A}]$ the (now supporting) $\mathcal{A}$ defeated.

Proof. (1) Given $\Lambda_{2n+1} = [\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{2n+1}]$, by Lemma 1.3.21 (1) we have $\Lambda'_{2n+1} = [\mathcal{A}_2, \ldots, \mathcal{A}_{2n+1}]$ is an arg. line. If $\mathcal{A}_{2n+1}$ is undefeated in Λ' we are done. Otherwise some $\mathcal{A}_{2n+2}$ exists with $\Lambda'_{2n+2} = [\mathcal{A}_2, \ldots, \mathcal{A}_{2n+1}, \mathcal{A}_{2n+2}]$ and $\mathcal{A}_{2n+2}$ evaluated as undefeated in Λ'_{2n+2}. Then $\Lambda_{2n+2} = \Lambda_{2n+1} {}^\frown[\mathcal{A}_{2n+2}]$ is an arg. line, and $\mathcal{A}_{2n+2}$ must be defeated there, since $\mathcal{A}_{2n+1}$ is undefeated. So some $\mathcal{A}_{2n+3}$ exists such that $\Lambda_{2n+3} = \Lambda_{2n+2} {}^\frown[\mathcal{A}_{2n+3}]$ is an arg. line and $\mathcal{A}_{2n+3}$ is undefeated there. This procedure can be repeated, as before generating an infinite sequence of increasing argumentation lines, which is impossible.

(2) The proof is analogous: let $\Lambda_{2n} = [\mathcal{A}_1, \ldots, \mathcal{A}_{2n}]$ be an arg. line with $\mathcal{A}_{2n}$ defeated. Then some $\mathcal{A}_{2n+1}$ exists with $\Lambda_{2n+1} = \Lambda_{2n} {}^\frown[\mathcal{A}_{2n+1}]$ and $\mathcal{A}_{2n+1}$ undefeated there. On the other hand, clearly $\Lambda'_{2n} = [\mathcal{A}_2, \ldots, \mathcal{A}_{2n}]$ is an arg. line, so if $\mathcal{A}_{2n}$ is defeated there we are done. Otherwise, some $\mathcal{A}_{2n+1}$ exists with $\Lambda'_{2n+1} = \Lambda'_{2n} {}^\frown[\mathcal{A}_{2n+1}]$ and $\mathcal{A}_{2n+1}$ defeated there. Then some $\mathcal{A}_{2n+2}$ exists with $\Lambda'_{2n+2} = \Lambda'_{2n+1} {}^\frown[\mathcal{A}_{2n+2}]$ and $\mathcal{A}_{2n+2}$ undefeated there. Then, $\mathcal{A}_{2n+2} = \Lambda_{2n+1} {}^\frown[\mathcal{A}_{2n+2}]$ is an arg. line. Since we had $\mathcal{A}_{2n+1}$ is undefeated, $\mathcal{A}_{2n+2}$ is defeated. This procedure can be repeated, again giving an infinite sequence of increasing arg. lines, which was shown to be impossible. □

Lemma. 1.5.3 Let $\mathcal{A}, \mathcal{B}$ be two arguments in some mutex program (Π, Δ). Let $\mathcal{A}, \mathcal{B}$ be arguments attacking each other (resp. with conclusions $\langle \ell, t \rangle$ and $\langle \sim\ell, t \rangle$), and such that $\mathcal{A} \cap \mathcal{B}$ is an argument and $\mathcal{B} \setminus \mathcal{A} \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$. Then,

(1) $\mathcal{A} \setminus \mathcal{B} \not\subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$

(2) $\mathsf{base}(\mathcal{A}) = \mathsf{base}(\mathcal{B})$

Proof. For claim (1): on the one hand, since $\mathcal{A} \cap \mathcal{B}$ is an argument, it has a unique conclusion $\mathsf{concl}(\mathcal{A} \cap \mathcal{B})$. Moreover, each rule δ in $\mathcal{B} \setminus \mathcal{A}$ is in $\Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$, so δ has a unique literal in $\mathsf{body}(\delta)$. Thus, $\mathcal{B} \setminus \mathcal{A}$ consists of a sequence of rules $\langle \delta_1^{\mathcal{B}}, \ldots, \delta_k^{\mathcal{B}} \rangle$ from $\Delta_p \cup \Pi_{\mathbf{M}}$; that is, with $\mathsf{body}(\delta_{i+1}) = \{\mathsf{head}(\delta_i^{\mathcal{B}})\}$. Assume now the contrary of claim (1), so we obtain a similar fact for $\mathcal{A} \setminus \mathcal{B}$, i.e. a sequence $\langle \delta_1^{\mathcal{A}}, \ldots, \delta_m^{\mathcal{A}} \rangle$ from $\Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$. Now, it is obvious that using the literal $\mathsf{concl}(\mathcal{A} \cap \mathcal{B})$ shared by $\mathcal{A}, \mathcal{B}$ and arbitrary rules from Δ_{p} and $\Pi_{\mathbf{M}}$ one cannot built a conflict. To see this, let $\mathsf{concl}(\mathcal{A} \cap \mathcal{B}) = \langle \ell, t \rangle$. If this is a negative literal $\langle {\sim}p, t \rangle$, then the rules $\delta_i^{\mathcal{A}}, \delta_j^{\mathcal{B}}$ can only be persistence rules $\delta_{{\sim}p}(t') \in \Delta_{\mathsf{p}}$. In this case, $\mathsf{concl}(\mathcal{A}) = \langle {\sim}p, \cdot \rangle = \mathsf{concl}(\mathcal{B})$. If $\langle \ell, t \rangle$ is a positive literal, say $\langle p, t \rangle$, then each sequence $\langle \delta_i^{\mathcal{A}} \rangle_{1 \leq i \leq m}$ and $\langle \delta_j^{\mathcal{B}} \rangle_{1 \leq j \leq k}$ contains at most a rule $\delta_i^{\mathcal{A}}, \delta_j^{\mathcal{B}}$ from $\Pi_{\mathbf{M}}$. The reason is that after such a rule the literal is negative (even after applying persistence) and so no other rule from $\Pi_{\mathbf{M}}$ will apply. Thus, the presumed "conflict" between $\mathsf{concl}(\mathcal{A})$ and $\mathsf{concl}(\mathcal{B})$ will be of the form: $\langle p, \cdot \rangle$ vs $\langle p, \cdot \rangle$, or $\langle p, \cdot \rangle$ vs $\langle {\sim}q, \cdot \rangle$ (for $p, q \in X \in \mathbf{M}$) or $\langle {\sim}r, \cdot \rangle$ vs $\langle {\sim}q, \cdot \rangle$ (for $q, r \in X \in \mathbf{M}$). In no such case, the arguments $\mathcal{A}$ and $\mathcal{B}$ can attack each other (contradiction).

For claim (2), since $\mathcal{A} \cap \mathcal{B}$ is an argument and both $\mathcal{A} \setminus \mathcal{B}$ and $\mathcal{B} \setminus \mathcal{A}$ consist of a sequence of rules with one literal in their body, these rules $\langle \delta_i^{\mathcal{A}} \rangle_{1 \leq i \leq m}$ and $\langle \delta_j^{\mathcal{B}} \rangle_{1 \leq j \leq k}$ must ultimately be based on $\mathsf{concl}(\mathcal{A} \cap \mathcal{B})$. It is straightforward to conclude from this that $\mathsf{base}(\mathcal{A}) = \mathsf{base}(\mathcal{A} \cap \mathcal{B}) = \mathsf{base}(\mathcal{B})$. □

Lemma. 1.5.4 Definitions 1.3.15 and 1.5.2 are equivalent for mutex programs.

Proof. Only the equivalence between each version of the second criterion needs to be checked.

($\Rightarrow$) The 1st condition from Def. 1.3.15 is exactly the same than the 1st condition of Def. 1.5.2, so we are done. On the other hand, the 2nd condition from Def. 1.3.15 states that $\mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi$. In order to show the 2nd condition from Def. 1.5.2, namely $\mathcal{B} \setminus \mathcal{A}_1 \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$, assume the contrary. Let then $\langle \ell_0, t_0 \rangle \in \Pi_{\mathsf{f}}$ be a strict fact in $\mathcal{B} \setminus \mathcal{A}_1$. Since $\mathcal{A}_1 \cap \mathcal{B}$ is non-empty (it is an argument), $\mathsf{concl}(\mathcal{B}) \neq \langle \ell_0, t_0 \rangle$. Moreover, by the $\subseteq$-minimality of $\mathcal{B}$, the existence of $\langle \ell_0, t_0 \rangle$ implies $\mathsf{concl}(\mathcal{A}_1 \cap \mathcal{B}) \neq \mathsf{concl}(\mathcal{B})$. Again by the $\subseteq$-minimaility of $\mathcal{B}$, the latter fact implies that eacg $\mathsf{concl}(\mathcal{A}_1 \cap \mathcal{B})$ and $\langle \ell_0, t_0 \rangle$ is in the body of some corresponding rule in $\mathcal{B} \setminus \mathcal{A}_1$. But by assumption, these rules from $\Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$ only take one literal at each body, so either these rules in $\mathcal{B} \setminus \mathcal{A}_1$ are ultimately based on $\mathsf{concl}(\mathcal{A}_1 \cap \mathcal{B})$ or on $\langle \ell_0, t_0 \rangle$. In either case $\mathcal{B}$ is not $\subseteq$-minimal.

($\Leftarrow$) For this direction, notice that the 1st and 2nd conditions from Def. 1.3.15 are satisfied. In order to check the 3rd condition $(\mathcal{B} \setminus \mathcal{A}_1) \cap \Delta_{\mathsf{p}} \neq \varnothing$, assume instead that $\mathcal{B} \setminus \mathcal{A}_1 \subseteq \Pi_{\mathbf{M}}$. Then, $\mathcal{B} \setminus \mathcal{A}_1$ must consist of a single mutex rule $\delta^{\mathbf{M}} = \langle {\sim}q, t \rangle \leftarrow \langle p, t \rangle$. This implies that $\mathsf{concl}(\mathcal{A}_1 \cap \mathcal{B}) = \langle p, t \rangle$. But this contradicts the fact that this conclusion must be of the form $\langle \cdot, t' \rangle$ with $t' < t$. The 4rth condition simply follows from Lemma 1.5.3 (1). And the 5th condition is Lemma 1.5.3 (b). □

Lemma. 1.5.7 Let $\mathcal{A}, \mathcal{B}$ be arguments in some mutex program (Π, Δ), with $\{\mathsf{concl}(\mathcal{A}), \mathsf{concl}(\mathcal{B})\} \subseteq X \in \mathbf{M}$. Define

$$\begin{array}{lcll} \mathcal{A}^+ & = & \mathcal{A} \cup \{\delta_{\mathcal{A}^+}\} & \text{with } \delta_{\mathcal{A}^+} = \sim\mathsf{concl}(\mathcal{B}) \leftarrow \mathsf{concl}(\mathcal{A}) \\ \mathcal{B}^+ & = & \mathcal{B} \cup \{\delta_{\mathcal{B}^+}\} & \text{with } \delta_{\mathcal{B}^+} = \sim\mathsf{concl}(\mathcal{A}) \leftarrow \mathsf{concl}(\mathcal{B}) \end{array}$$

Assume that $\mathcal{A}^+$ and $\mathcal{B}^+$ are arguments in (Π, Δ). Then the following equivalences hold:

$$\mathcal{A}^+ \succ \mathcal{B} \text{ iff } \mathcal{A} \succ \mathcal{B}^+ \qquad \text{and} \qquad \mathcal{A}^+ \prec\!\!\succ \mathcal{B} \text{ iff } \mathcal{A} \prec\!\!\succ \mathcal{B}^+$$

Proof. For the first equivalence, we only show the $\Rightarrow$ direction, the other direction being analogous. Recall that $\mathcal{A}^+ \succ \mathcal{B}$ is defined by the disjunction: either $\mathsf{base}(\mathcal{A}^+) \supsetneq \mathsf{base}(\mathcal{B})$ or $\mathcal{A}^+ \cap \mathcal{B}$ is an argument and $\mathcal{B} \setminus \mathcal{A}^+ \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$.

(Case $\mathsf{base}(\mathcal{A}^+) \supsetneq \mathsf{base}(\mathcal{B})$.) Then, since obviously $\mathsf{base}(\mathcal{A}) = \mathsf{base}(\mathcal{A}^+)$ and $\mathsf{base}(\mathcal{B}^+) = \mathsf{base}(\mathcal{B})$, we have that this case is equivalent to $\mathsf{base}(\mathcal{A}) \supsetneq \mathsf{base}(\mathcal{B}^+)$, which implies $\mathcal{A} \succ \mathcal{B}^+$ so we are done.

(Case $\mathcal{A}^+ \cap \mathcal{B}$ is an argument and $\mathcal{B} \setminus \mathcal{A}^+ \subseteq \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}}$.) On the one hand, $\mathcal{A} \cap \mathcal{B}^+$ is an argument as well. The reason is that $\delta_{\mathcal{B}^+} \notin \mathcal{A}$, since otherwise the fact that $\mathcal{A}$ is an argument ($\subseteq$-minimal w.r.t. derivation) would imply that $\mathsf{concl}(\mathcal{A}), \mathsf{concl}(\mathcal{B}^+) \in \mathsf{literals}(\mathcal{A})$; but this pair is $\mathsf{concl}(\mathcal{A}), \sim\mathsf{concl}(\mathcal{A})$, thus making $\mathcal{A}$ inconsistent (in itself, and so with Π), contradicting that $\mathcal{A}$ is an argument. Thus, we conclude $\delta_{\mathcal{B}^+} \notin \mathcal{A}$ and hence $\mathcal{A} \cap \mathcal{B}^+ = \mathcal{A} \cap \mathcal{B}$. Reasoning similarly, we obtain that $\delta_{\mathcal{A}^+} \notin \mathcal{B}$, so $\mathcal{A}^+ \cap \mathcal{B} = \mathcal{A} \cap \mathcal{B}$. Combining the last two identities, we conclude that $\mathcal{A} \cap \mathcal{B}^+ = \mathcal{A}^+ \cap \mathcal{B}$. Since the former is an argument, so is the latter.

On the other hand, we have

$$\begin{array}{rcll} \mathcal{B} \setminus \mathcal{A}^+ & \subseteq & \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \\ (\mathcal{B} \setminus \mathcal{A}^+) \cup (\mathcal{A}^+ \cap \mathcal{B}) & \subseteq & (\mathcal{A}^+ \cap \mathcal{B}) \cup \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \\ \mathcal{B} & \subseteq & (\mathcal{A}^+ \cap \mathcal{B}) \cup \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \\ \mathcal{B} & \subseteq & (\mathcal{A} \cap \mathcal{B}) \cup \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \text{since } \delta_{\mathcal{A}^+} \in \Pi_{\mathbf{M}} \\ \mathcal{B}^+ & \subseteq & (\mathcal{A} \cap \mathcal{B}) \cup \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \text{since } \delta_{\mathcal{B}^+} \in \Pi_{\mathbf{M}} \\ \mathcal{B}^+ & \subseteq & (\mathcal{A} \cap \mathcal{B}^+) \cup \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \text{since } \mathcal{B} \subseteq \mathcal{B}^+ \\ (\mathcal{B}^+ \setminus \mathcal{A}) \cup (\mathcal{A} \cap \mathcal{B}^+) & \subseteq & (\mathcal{A} \cap \mathcal{B}^+) \cup \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \\ \mathcal{B}^+ \setminus \mathcal{A} & \subseteq & \Delta_{\mathsf{p}} \cup \Pi_{\mathbf{M}} & \text{since } \mathcal{B}^+ \setminus \mathcal{A} \text{ and} \\ & & & \mathcal{A} \cap \mathcal{B}^+ \text{ are disjoint} \end{array}$$

This completes the proof that $\mathcal{A}^+ \succ \mathcal{B} \Rightarrow \mathcal{A} \succ \mathcal{B}^+$.

The second equivalence in the Lemma claim follows from the previous proof:

$$\begin{array}{rcl} \mathcal{A}^+ \succ \mathcal{B} & \Leftrightarrow & \mathcal{A}^+ \not\prec \mathcal{B} \text{ and } \mathcal{A}^+ \not\succ \mathcal{B} \\ \Leftrightarrow \quad \mathcal{A} \not\prec \mathcal{B}^+ \text{ and } \mathcal{A} \not\succ \mathcal{B}^+ & \Leftrightarrow & \mathcal{A} \prec\!\!\succ \mathcal{B} \end{array}$$

□

Chapter 2

A Planning System based on t-DeLP for centralized planning

2.1 Introduction

In the present chapter, we study a temporal planning system built on the t-DeLP logic programming framework from Chapter 1. The present focus is on centralized planning, leaving distributed algorithms for collaborative planning for Chapter 3.

A centralized approach to planning assumes a unique planner (algorithm), who is assigned the task of finding a joint plan for a set of executing agents at her disposal. This planner is endowed with some goals and some knowledge of (at least) the current state and the agents' possible actions. The planner makes use of this knowledge both to build plans in a stepwise fashion and to evaluate them (by computing the consequences of adding a new action to some plan under consideration).

Although this chapter is focused on Breadth First Search (BFS) as the search method for the space of states, the results easily generalize to other search methods in the literature. The BFS algorithm is studied for both directions of incremental plan search: a forward approach (a search from the initial state to some goal state), and a backward approach (a search from the goals to the initial state). After studying the basic properties of BFS for the relatively simple case of forward planning, we devote the remaining of the chapter to the more complex case of backward planning. In either case –a forward or backward approach–, the BFS algorithm is shown to be sound and complete for the t-DeLP planning system.

The resulting planning system, as can be expected, inherits the defeasible reasoning abilities of t-DeLP, and combines it with dynamic reasoning about the

execution of actions. All this is formally studied as a t-DeLP notion of a state transition system. Before proceeding with the motivation for a t-DeLP based approach to temporal planning, let us informally outline these states transition systems. Essentially, a state transition system is a function on states and actions (or events)

$$\text{states} \times \text{actions} \longrightarrow \text{states}$$

This function computes the new state that results after executing an action in a given state. In temporal planning, a state usually refers to a world-line, i.e. a set of descriptions of the scenario at different instants.

Actions, for the sake of simplicity, are represented here by a simple model of temporal actions from the literature on temporal planning [66]. [1] As usual, temporal actions in t-DeLP are sets of preconditions and (direct) effects; what makes them temporal is the fact that these sets consist of temporal literals. Thus, a t-DeLP temporal action in a plan is automatically scheduled for the time interval defined by its preconditions and its effects.

In the case of states, a natural candidate in t-DeLP logic programming for this notion of world-line is *the output* $\mathsf{warr}(\Pi, \Delta)$ *of a logical program* (Π, Δ). For technical reasons, though, an action will be seen as a map from logical programs to logical programs.

$$\begin{array}{ccccc} (\Pi, \Delta) & \times & \mathsf{action} & \longmapsto & (\Pi', \Delta) \\ \Downarrow & & & & \Downarrow \\ \mathsf{warr}(\Pi, \Delta) & & & & \mathsf{warr}(\Pi', \Delta) \end{array}$$

The updated program (Π', Δ) will incorporate the direct effects of the (executable) action as new strict facts. The indirect effects of the action in this state, at its turn, are simply the warranted literals in this program which were not warranted in the former program. Recall that for a mutex program (Π, Δ), its defeasible logical closure $\mathsf{warr}(\Pi, \Delta)$ is a consistent set of literals, *viz.* Theorem 1.5.8. But for most of these programs, the corresponding sets of literals $\mathsf{warr}(\Pi, \Delta)$ will be incomplete, e.g. $\langle \ell, t \rangle, \langle \sim \ell, t \rangle \notin \mathsf{warr}(\Pi, \Delta)$. Thus, this definition of states as sets of the form $\mathsf{warr}(\Pi, \Delta)$ in the t-DeLP planning system makes them partial states, at best. Indeed, given the results of the previous chapter, we will require in the remaining of Part I that any logical program considered during the planning phase is a mutex program. In other words, that the space of plans is the set of mutex programs in a given t-DeLP language.

The advantages of the t-DeLP planning system over (classical or) temporal planning are exactly the same that t-DeLP has over temporal logic programming and other non-monotonic logics for temporal reasoning: a non-monotonic approach to deal with the classical representation problems; and the use of argumentation tools leading to a non-monotonic approach based on natural concepts. The use of t-DeLP as the underlying logic for a planning system, in particular, is useful for reasoning or planning with:

[1] Although the t-DeLP planning system (and its formal properties) seem to extend to richer representations of temporal actions, formal proofs for this must be left for future work.

- the indirect effects of actions: whether a literal ℓ is caused at t by an action, or not caused at all by it (i.e. the ramification problem; compare with temporal planners in the literature [66])
- the exact time an action effect will really occur, when this time partly depends on the environment (again compare with temporal planners [66] and also with synchronous argumentation systems [95], [50])
- qualifications on the preconditions of actions (the qualification problem)
- a formal notion of persistence of facts through time (compare with [14])
- a user-friendly causal modeling system, in the sense of naturally representing naive descriptions of causal domains (compare with e.g. defeasible temporal logics [72])

Structure of the Chapter.

This chapter is structured as follows. First in Section2.2, we motivate the present approach with a comparison with standard (classical or temporal) planning systems in terms of reasoning power. Second, in Section 2.2.1, we briefly list some simplifying assumptions on the representation of temporal actions and agents in t-DeLP; these allow for more compact definitions and proofs for the present task, while sufficing to illustrate the proposed common-sense notions of action, causation and persistence. Then, in Section 2.3 we define the update function for a state transition system based on t-DeLP. After this, we briefly study in Section 2.4 the easy case for forward planning using Breadth First Search. This algorithm is shown to be sound and complete. Finally, in Section2.5, we define the space of states for backward planning, and the Breadth First Search algorithm for centralized multi-agent planning in t-DeLP. We also prove that this planning algorithm is correct and complete.

2.2 Representing actions and indirect effects in planning.

In order to motivate the present approach, we briefly review some how actions are modeled in standard planning systems, and try to illustrate the relative limits of these models in terms of reasoning about actions. As Chapter 1 might suggest, these limitations originate in the use of simple monotonic logics and are inherited by planners based on them. The comparison is made in terms of classical planning actions, since the (quantitative) temporal aspects of actions in temporal planning do not make a difference for the present discussion.

A general assumption on the representation of actions in most planning systems (be it for actions with deterministic, conditional or disjunctive effects) is that the action encapsulates all the possible effects of its execution. This

makes the resulting state transition systems simple enough to prevent the frame problem to occur. For example, the execution of an action without conditional effects simply is simply the state defined by the actions effects and the previous state facts which are not the negation of some action effect are automatically preserved.

This simple model of actions, of course, cannot model that some propositions jointly imply a proposition. This is modeled by considering conditional effects, similar to the domain-specific rules used by t-DeLP

$$\underbrace{fact_1, \ldots, fact_n}_{\text{conditions}} \quad \blacktriangleright \quad \underbrace{fact}_{\text{effect}}$$

which impose further conditions on the preservation of previous facts: it must neither contradict any action effect nor any conditional effect whose conditions hold.

Example 2.2.1 (Door Opening)**.** Suppose one is to give a formal representation of different actions for opening a door. Say a *simple door* might just be opened by pushing it. A *complex door*, at its turn, can only be opened by pushing it while turning the doorknob (in particular, by pushing and turning the doorknob).

The problem of classical planning is that it cannot model concurrent actions (push and turn) in a modular way. The only possibility is to introduce a new action in the planning domain for each combination of actions which can be concurrently executed.

Example 2.2.2 (Door Opening; Classical Planning)**.** For the previous example, let us abstract from some obvious preconditions (that the door is closed, the agent is in front of it, and so on). Classical planning would model the previous door-opening actions as follows:

action	preconditions	effects
push	$simple(door)$	$open(door)$
push&turn	–	$open(door)$

As mentioned in [66], an explicit representation of concurrency like that of Example 2.2.2 is less expressive than one in which joint effects of concurrent or interfering actions, like temporal planning. In temporal planners, two actions are concurrent if they are scheduled for the same time. Concurrent actions might have non-additive effects, i.e. effects which are not in the description of either action. The exclusive effects of the joint action can be represented by conditional effects, e.g. (ce2) below.

Example 2.2.3 (Door Opening; Temporal Planning)**.** The same example 2.2.1 can be modeled in the language of a temporal planner as follows:

action	preconditions	effects
push	–	$pushed(door)$
turn	–	$turned(doorknob)$

cond. effect	conditions		effect
(ce1)	$\left\{\begin{array}{c} pushed(door), \\ simple(door) \end{array}\right\}$	▶	$open(door)$
(ce2)	$\left\{\begin{array}{c} pushed(door), \\ turned(doorknob) \end{array}\right\}$	▶	$open(door)$

Now, suppose we want to model an exception to the success of these actions, e.g. that the door is obstructed. One might simply redefine these actions by adding as a precondition that the exception does not occur. Or, following the previous example, one can (non-modularly) be modeled by rewriting the former conditional effects into (ce1*)-(ce2*) and adding a new set (ce3)-(ce4)

Example 2.2.4 (Door Opening; Temporal Planning, cont'd)**.** The handling of exceptions, e.g. *the door is obstructed*, can be done as follows.

(ce1*)
$$\left\{\begin{array}{c} pushed(door), \\ simple(door), \\ \sim obstructed(door) \end{array}\right\} \blacktriangleright open(door)$$

(ce3)
$$\left\{\begin{array}{c} pushed(door), \\ simple(door), \\ obstructed(door) \end{array}\right\} \blacktriangleright \sim open(door)$$

(ce2*)
$$\left\{\begin{array}{c} pushed(door), \\ turned(doorknob), \\ \sim obstructed(door) \end{array}\right\} \blacktriangleright open(door)$$

(ce4)
$$\left\{\begin{array}{c} pushed(door), \\ turned(doorknob), \\ obstructed(door) \end{array}\right\} \blacktriangleright \sim open(door)$$

Second- or higher-order exceptions (i.e. exceptions to exceptions, and so on) add a new level of increase in the size of the planning domain. For example, assume that strong agents can open both obstructed and unobstructed doors. Then Example 2.2.3 would need eight rules to take this into account. Each additional exception to be incorporated seems to require: (a) rewriting of the relevant rules (with an additional condition that the exception does not occur); and (b) the addition of a new set of the same size (for the case where the exception occurs and then the effect fails). In contrast, in t-DeLP the incorporation of new exceptions need not affect the previous representations.

Example 2.2.5 (Door Opening; t-DeLP planning)**.** Following Example 2.2.1, the addition below of δ_4, δ_5 to the set $\{\delta_1, \delta_2, \delta_3\}$ is done without modifying the elements of this set. (Note we follow the previous examples with a left-to-right representation of defeasible rules. Again, time or temporal literals are not explicitly represented. The set of actions is the same than in Example 2.2.4.)

defeasible rule	body		head
δ_1	$\{pushed(door)\}$	$\succ$	$open(door)$
δ_2	$\left\{\begin{array}{c} pushed(door), \\ complex(door) \end{array}\right\}$	$\succ$	$\sim open(door)$
δ_3	$\left\{\begin{array}{c} pushed(door), \\ turned(doorknob), \\ complex(door) \end{array}\right\}$	$\succ$	$open(door)$
δ_4	$\{obstructed(door)\}$	$\succ$	$\sim open(door)$
δ_5	$\left\{\begin{array}{c} obstructed(door), \\ strong(agent), \\ pushed(door) \end{array}\right\}$	$\succ$	$open(door)$

Using the tools from the previous Chapter 1 and Section 2.3 it can be shown that t-DeLP warrant captures the expected effects in each possible combination of the above described exceptions.

In summary, the main difference between defeasible rules in t-DeLP and conditional effects in classical or temporal planners is that the latter cannot work with

- an "inconsistent" set of conditional effects is not allowed, i.e. one cannot consider $\{p, \ldots, p'\} \blacktriangleright q$ and $\{r, \ldots, r'\} \blacktriangleright \sim q$ as conditional effects for a given action e, when $p, \ldots, p', r, \ldots, r'$ are jointly consistent
- (for many planners) indirect effects derivable in a finite number of steps, i.e. nested conditional effects.

Non-monotonic reasoning, and defeasible argumentation in particular, offers a solution to these issues. The general idea, as seen in Example 2.2.1 then to represent a real-world action by splitting it into: (i) a temporal planning action, encapsulating its direct, putative and incontestable effect(s); and (ii) a set of temporal defeasible rules, which combine with these direct effects and other external facts into arguments. Although the above example illustrates the use of indirect effects, qualifying the preconditions of an action with the help of defeasible rules is also possible using similar ideas.

2.2.1 A simple model for temporal actions in t-DeLP

In the present and the next chapter, we will assume certain simplified model of temporal action. These assumptions can be informally presented as follows. An action e has a unique effect, denoted μ_e (or $\langle \mu_\mathsf{e}, t_\mathsf{e} \rangle$). The effect μ_e is exclusive to action e (not found in nature, or other actions) and cannot be contradicted once it is made true (not even by strict facts or mutex rules). The effect μ_e can be simply read as *action* e *was just executed in* t_e. Thus we use a special symbol μ_e to denote the effect of an action e, even if these μ symbols are just propositional variables in Var.

A detailed list of these assumptions for actions, and additional assumptions on executing agents, is given in Fig. 2.1. The purpose of all these assumptions is to simplify the definitions and the proofs of the Chapter 2. Many of them can actually be dropped, allowing for more general notions of actions, as found in the temporal planning literature. In most cases, the current proofs can easily be adapted to these more expressive representations.

	For any actions e, f and each rule or fact $\delta \in \Pi \cup \Delta$,
(Exclusivity)	$\mu_{\mathsf{e}} \neq \mu_{\mathsf{f}}, \sim\mu_{\mathsf{f}}$ and $\mu_{\mathsf{e}}, \sim\mu_{\mathsf{e}} \notin \bigcup \mathbf{M}$ $\mu_{\mathsf{e}} \neq \mathsf{head}(\delta)$ and $\sim\mu_{\mathsf{e}} \notin \mathsf{body}(\delta) \cup \{\mathsf{head}(\delta)\}$
(Simple prec.)	preconditions $\mathsf{pre}(\mathsf{e})$ need only be true at the start of the execution of e (not during part or all of it).
(Simple effect)	the effect $\mathsf{post}(\mathsf{e})$ is only strictly true (i.e. in Π) just after the execution of e, not during this execution.
(Simult. prec.)	the preconditions in $\mathsf{pre}(\mathsf{e})$ are about a single time-point t, where these must simultaneously hold: $\mathsf{pre}(\mathsf{e}) = \{\langle \ell, t\rangle, \ldots, \langle \ell', t\rangle\}$
(Future effects)	for any action e, we assume $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$ is to occur later than these preconditions (i.e. $t < t_{\mathsf{e}}$).
(Simple duration)	the duration of any action e is set to 1 time unit; that is, if the preconditions are $\mathsf{pre}(\mathsf{e}) = \{\langle \cdot, t\rangle, \ldots \langle \cdot, t\rangle\}$, the effect will be of the form $\langle \mu_{\mathsf{e}}, t+1\rangle$.
(Single-task agents)	the execution action e makes the executing agent or actuator a busy, e.g. during the interval $[t, t+1]$
(Enough agents)	the set of available agents is finite but sufficiently large for the planning problem at hand: if a solution with n agents simultaneously acting exists, then we were assuming from start that n agents exist

Figure 2.1: A list of assumptions on the multi-agent planning system in t-DeLP.

2.3 Basic concepts in t-DeLP multi-agent planning

We proceed with the basic definitions for centralized multi-agent planning systems based on t-DeLP. These are the notions of action, planning domain, forward plan, and the update or progression function.

In the particular case of actions, the same comments that we made when introducing defeasible rules in Ch. 1 apply here. Thus, in principle we would introduce actions $\mathsf{e} = (\mathsf{pre}(\mathsf{e}), \mathsf{post}(\mathsf{e}))$ as action schemas $\mathsf{e}_{\mathbf{t}}$, that is, with non-instantiated temporal literals: $\mathsf{pre}(\mathsf{e}_{\mathbf{t}}) = \{\langle \ell, \mathbf{t}\rangle, \langle \ell', \mathbf{t}\rangle, \ldots, \}$ and $\mathsf{post}(\mathsf{e}_{\mathbf{t}}) = \{\langle \mu_{\mathsf{e}}, \mathbf{t}+1\rangle\}$. And then consider particular instantiations, where $\mathbf{t}$ can take any value in ω; for example e_{10} would have as effect $\mathsf{post}(\mathsf{e}_{10}) = \{\langle \mu_{\mathsf{e}}, 11\rangle\}$. But in practice, since the goals in the planning domain are bounded, say, at t, we will define the set of actions A as the set of instantiations of actions $\mathsf{e}_{\mathbf{t}}$ with $\mathbf{t}$ ranging between 0 and t. Since a planning domain would only consider finitely-many action schemas, our set of (instantiated) actions A will be finite as well.

Definition 2.3.1 (Action. Executability.)**.** Let a language TLit of temporal literals be given. An *action* in TLit is a pair $\mathsf{e} = (\mathsf{pre}(\mathsf{e}), \mathsf{post}(\mathsf{e})) \subseteq \mathsf{TLit} \times \mathsf{TLit}$, where $\mathsf{pre}(\mathsf{e}) = \{\langle \ell, t\rangle, \ldots, \langle \ell', t\rangle\}$ is a consistent set of temporal literals and $\mathsf{post}(\mathsf{e}) = \{\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle\}$, with $t < t_{\mathsf{e}} = t+1$. The set $\mathsf{pre}(\mathsf{e})$ and the literal $\mathsf{post}(\mathsf{e})$ are called, resp., the *preconditions* and the *(direct) effect* of e. We will sometimes forget about the set $\mathsf{post}(\mathsf{e})$ and will identify $\mathsf{post}(\mathsf{e}) = \langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$ directly. We say an action e is *executable* in a t-DeLP program (Π, Δ) (also in the same language TLit) iff $\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\Pi, \Delta)$.

Given a set of agents (or actuators) $\mathsf{Ag} = \{a, b, \ldots\}$, we denote an action available to agent a by e_a.

Thus, in case several actions are executed simultaneously at t it does not matter the order in which we compute their update. The only requirement for plans is that concurrent actions are executed by a different agent (or actuator) each.

Definition 2.3.2 (Non-overlapping Actions)**.** We say a set of actions $A' \subseteq A$ is *non-overlapping for each agent* $a \in \mathsf{Ag}$ iff for any two actions of an agent in A', say $\mathsf{e}_a, \mathsf{f}_a \in A_a \cap A'$ the effect of e_a is to occur strictly before the preconditions of f, or viceversa.

An action e_a available to agent a can obviously require something from a as a precondition (e.g. that the agent is in some location). In order to simplify the notation for actions e_a, we will usually drop the sub-index a and leave it implicit which is the executing agent.

Definition 2.3.3 (Planning Domain)**.** For a fixed set of agents Ag and a language TLit of temporal literals, we define a *planning domain* as any triple of the form

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

where (Π, Δ) is a t-DeLP mutex program, with $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathsf{r}}$ as usual. The sets of temporal literals $\Pi_{\mathsf{f}}, G \subseteq \mathsf{TLit}$, denote, resp., the *strict facts* in the initial state and the *goals* considered by the planner; A is a set of actions for agents in Ag. The sets Π, Δ are assumed to satisfy the constraints about $\langle \mu_{\mathsf{e}}, \cdot \rangle$ and $\langle \sim\mu_{\mathsf{e}}, \cdot \rangle$ literals from Fig. 2.1.

In contrast to classical planning, our update function will be applied to t-DeLP programs (Π, Δ) rather than to states s. Indeed, a t-DeLP program (Π, Δ) can be identified with the partial state given by $\mathsf{warr}(\Pi, \Delta)$, which -in contrast to the classical notion- needs not be a maximally consistent sets of literals. After an action e is executed in a t-DeLP program (Π, Δ), we expand the set of strict facts in Π by adding each effect of e as a new strict fact. Let us remark two aspects:

(a) it does not make a difference whether the preconditions of e were warranted using strict information from Π alone, or using some defeasible undefeated argument: once we accept them as warranted, the effects become equally established as (future) facts;

(b) the orientation towards the future of actions and rules prevents undesired circularities, that might be read as time paradoxes: the execution of an action e might enforce an argument $\mathcal{A}$ whose conclusion contradicted the preconditions for e, making this executed action non-executable.

Definition 2.3.4 (Action Update)**.** Let (Π, Δ) be a t-DeLP program in some language TLit and e an action in the same language. We define the *update function* as a mapping between a pair (program, action) and a program:

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

Since each action e is defined with its own schedule (the temporal parameters $\langle \cdot, t \rangle$ in preconditions and effects), plans need not be explicitly defined as sequences of actions. Indeed, we will use a more flexible representation of a plan, namely a set of actions. Let us formally justify the claim that both the sequence- and the set-based representations are equivalent. For this, a first auxiliary result states that adding a new strict fact $\langle \cdot, t' \rangle$ to some t-DeLP program (Π, Δ) does not change the warrant status of previous literals $\langle \ell, t \rangle$, i.e. with $t < t'$.

Lemma 2.3.5. *Let (Π, Δ) be a t-DeLP program and let $\langle \ell, t \rangle, \langle \ell', t' \rangle$ be arbitrary literals consistent with Π. Then, $t < t'$ implies*

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

Proof. The assumption $t < t'$ implies that any argument $\mathcal{D}$ for some literal $\langle \cdot, t \rangle$ with $t < t'$ in $(\Pi \cup \{\langle \ell', t' \rangle\}, \Delta)$ is also an argument in (Π, Δ), since by definition of Δ, the argument $\mathcal{D}$ cannot be based on $\langle \ell', t' \rangle$. And viceversa, each argument $\mathcal{D}$ in (Π, Δ) for $\langle \cdot, t \rangle$ is clearly an argument in $(\Pi \cup \{\langle \ell', t' \rangle\}, \Delta)$, since

the consistency of $\mathsf{literals}(\mathcal{D}) \cup \Pi$ will be preserved into $\mathsf{literals}(\mathcal{D}) \cup \Pi \cup \mathsf{post}(\mathsf{f})$. Thus, the arguments for such literals $\langle \cdot, t \rangle$ with $t < t'$ are exactly the same between these two t-DeLP programs.

Now, let $\mathcal{A}$ be an arbitrary argument for $\langle \ell, t \rangle$. Since $t < t'$, the latter claim implies that $\mathcal{T}_{(\Pi \cup \{\langle \ell', t' \rangle\}, \Delta)}(\mathcal{A})$ is identical to $\mathcal{T}_{(\Pi, \Delta)}(\mathcal{A})$. Hence, the undefeated status of $\mathcal{A}$ is exactly the same in these two trees. Since $\mathcal{A}$ was arbitrary, this implies that the $\langle \ell, t \rangle$ is in $\mathsf{warr}((\Pi \cup \{\langle \ell', t' \rangle\}, \Delta)$ iff it is in $\mathsf{warr}(\Pi, \Delta)$. □

Using this Lemma, we show next that for any two actions planned to be simultaneously executed, it does not matter the particular order in which their effects are computed.

Lemma 2.3.6. *Let (Π, Δ) be a t-DeLP program and e, f a pair of actions. If these actions are simultaneous: $\mathsf{pre}(\mathsf{e}) = \{\langle \ell, t \rangle, \ldots\}$ and $\mathsf{pre}(\mathsf{f}) = \{\langle \ell', t \rangle, \ldots\}$, then*

$$((\Pi, \Delta) \diamond \mathsf{e}) \diamond \mathsf{f} = ((\Pi, \Delta) \diamond \mathsf{f}) \diamond \mathsf{e}$$

Proof. Since $\mathsf{post}(\mathsf{e}) = \langle \mu_{\mathsf{e}}, t+1 \rangle$ and $\mathsf{post}(\mathsf{f}) = \langle \mu_{\mathsf{f}}, t+1 \rangle$, we can apply Lemma 2.3.5 to the case $t' = t + 1 (= t_{\mathsf{e}} = t_{\mathsf{f}})$ to obtain

$$\begin{array}{lcl} (\star) \quad \mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) & \Leftrightarrow & \mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\Pi, \Delta) \\ (\star 2) \quad \mathsf{pre}(\mathsf{f}) \subseteq \mathsf{warr}(\Pi \cup \mathsf{post}(\mathsf{e}), \Delta) & \Leftrightarrow & \mathsf{pre}(\mathsf{f}) \subseteq \mathsf{warr}(\Pi, \Delta) \end{array}$$

We will use these facts in the following proof by cases.

(Case $(\Pi, \Delta) \diamond \mathsf{e} = (\Pi, \Delta) = (\Pi, \Delta) \diamond \mathsf{f}$)

$$\begin{array}{rcl} ((\Pi, \Delta) \diamond \mathsf{e}) \diamond \mathsf{f} & = & (\Pi, \Delta) \diamond \mathsf{f} \\ & = & (\Pi, \Delta) \\ & = & (\Pi, \Delta) \diamond \mathsf{e} \\ & = & ((\Pi, \Delta) \diamond \mathsf{f}) \diamond \mathsf{e} \end{array}$$

(Case $(\Pi, \Delta) \diamond \mathsf{e} = (\Pi, \Delta)$ and $(\Pi, \Delta) \diamond \mathsf{f} = (\Pi \cup \mathsf{post}(\mathsf{f}), \Delta)$.)

$$\begin{array}{rcll} & ((\Pi, \Delta) \diamond \mathsf{f}) \diamond \mathsf{e} & & \\ = & (\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) \diamond \mathsf{e} & & \text{(Case assumption)} \\ = & \begin{cases} (\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) & \text{if } \mathsf{pre}(\mathsf{e}) \not\subseteq \mathsf{warr}(\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) \\ (\Pi \cup \mathsf{post}(\mathsf{f}) \cup \mathsf{post}(\mathsf{e}), \Delta) & \text{if } \mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) \end{cases} & \\ = & \begin{cases} (\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) & \text{if } \mathsf{pre}(\mathsf{e}) \not\subseteq \mathsf{warr}(\Pi, \Delta) \\ (\Pi \cup \mathsf{post}(\mathsf{f}) \cup \mathsf{post}(\mathsf{e}), \Delta) & \text{if } \mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\Pi, \Delta) \end{cases} & \text{(by } \star) \\ = & (\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) & & \text{(Case assumption)} \\ = & (\Pi, \Delta) \diamond \mathsf{f} & & \text{(Case assumption)} \\ = & ((\Pi, \Delta) \diamond \mathsf{e}) \diamond \mathsf{f}) & & \text{(Case assumption)} \end{array}$$

(Case $(\Pi, \Delta) \diamond \mathsf{e} = (\Pi \cup \mathsf{post}(\mathsf{e}), \Delta)$ and $(\Pi, \Delta) \diamond \mathsf{f} = (\Pi, \Delta)$)
The proof is analogous to the previous case, just replace e and f by each other and use $(\star 2)$ instead of $(\star)$.

(Case $(\Pi, \Delta) \diamond \mathsf{e} = (\Pi \cup \mathsf{post}(\mathsf{e}), \Delta)$ and $(\Pi, \Delta) \diamond \mathsf{f} = (\Pi \cup \mathsf{post}(\mathsf{f}), \Delta)$)
We denote each assumption in this case by

$$\begin{array}{ll} \text{(i)} & (\Pi, \Delta) \diamond \mathsf{e} = (\Pi \cup \mathsf{post}(\mathsf{e}), \Delta) \\ \text{(ii)} & (\Pi, \Delta) \diamond \mathsf{f} = (\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) \end{array}$$

Then we reason as follows:

$$\begin{array}{lll} & ((\Pi, \Delta) \diamond \mathsf{e}) \diamond \mathsf{f} & \\ = & ((\Pi \cup \mathsf{post}(\mathsf{e}), \Delta) \diamond \mathsf{f} & \text{(Case Assumption (i))} \\ = & ((\Pi \cup \mathsf{post}(\mathsf{e}) \cup \mathsf{post}(\mathsf{f}), \Delta) & \text{(by (ii) and } (\star 2)) \\ = & ((\Pi \cup \mathsf{post}(\mathsf{f}), \Delta) \diamond \mathsf{e} & \text{(by (i) and } (\star)) \\ = & ((\Pi, \Delta) \diamond \mathsf{f}) \diamond \mathsf{e} & \text{(by (ii))} \end{array}$$

□

Thus, in case several actions are executed simultaneously at t, it does not matter the order in which we compute their update. Using Lemmas 2.3.5 and 2.3.6, we can define an update by a set of scheduled actions, rather than, say, requiring these actions to be ordered in a sequence according to the schedule.

Definition 2.3.7 (Plan update)**.** Let (Π, Δ) be a t-DeLP program and $\{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$ a set of temporal actions with $\mathsf{pre}(\mathsf{e}_i) = \{\langle \ell, t_i \rangle, \ldots\}$. We define the *update* of a t-DeLP program by a set of actions as follows:

$$\begin{array}{lll} (\Pi, \Delta) \diamond \varnothing & = & (\Pi, \Delta) \\ (\Pi, \Delta) \diamond \{\mathsf{e}_1, \ldots, \mathsf{e}_n\} & = & ((\Pi, \Delta) \diamond \mathsf{e}_i) \diamond \{\mathsf{e}_1, \ldots, \mathsf{e}_{i-1}, \mathsf{e}_{i+1}, \ldots, \mathsf{e}_n\} \\ & & \text{where } t_i \leq t_j \text{ for any } 1 \leq j \leq n \end{array}$$

Finally, let us define what a solution to some planning domain is. A solution is just a plan that makes the goals true after the execution.

Definition 2.3.8 (Solution)**.** We say a set of actions $A' \subseteq A$ is a *solution* for $\mathbb{M} = ((\Pi, \Delta), A, G)$ iff

$$G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A')$$

and A' is non-overlapping for each agent $a \in \mathsf{Ag}$. In the general case where a plan need not be defined as a set of actions, we will also say that a plan π is a *solution* for $\mathbb{M}$ iff its set of actions $A(\pi)$ is a solution for $\mathbb{M}$.

Let us illustrate these concepts in the next Example 2.3.9 and Figure 2.2. (Unlike Example 2.2.5, all the temporal elements are made explicit here).

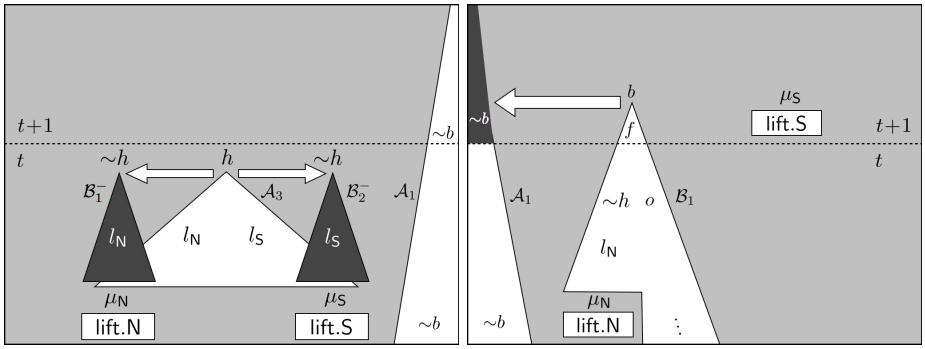


Figure 2.2: (Left) A representation of defeated arguments (dark triangles), capturing the effects of either single lifting action. The central argument, representing the simultaneous lifting of both sides of the table at time t, defeats each of those. The rightmost argument step $\mathcal{A}_1$ is unattacked. (Right) A representation of a non-simultaneous lifting of both sides of the table (at t, and resp. at $t+1$).

Example 2.3.9 (Table Lifting)**.** Let us suppose that the planner, endowed with two agents $\mathsf{Ag} = \{a_1, a_2\}$, wants some table to be lifted, without breaking a vase which lies on the table. The table has two sides (north and south), which can be lifted by either action, say $\mathsf{lift.N} \in A_{a_1}$ and $\mathsf{lift.S} \in A_{a_2}$. Particular instances of these actions at some time t are denoted $\mathsf{lift.N}^t, \mathsf{lift.S}^t$. This t denotes that the preconditions are of the form $\langle \cdot, t \rangle$. Consider the next abbreviations for facts:

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

Next, we define the following goals $G = \{\langle l_{\mathsf{N}}, 10\rangle, \langle l_{\mathsf{S}}, 10\rangle, \langle \sim b, 10\rangle\}$, the initial facts $\Pi_{a_1} = \Pi_{a_2} = \{\langle \sim b, 0\rangle, \langle h, 0\rangle \langle \sim l_{\mathsf{N}}, 0\rangle \langle \sim l_{\mathsf{S}}, 0\rangle, \langle o, 0\rangle\}$, and the set of defeasible rules Δ:

$$
\begin{array}{llllllll}
\delta_1: & \langle \sim h, t\rangle & \prec & \langle l_{\mathsf{N}}, t\rangle & \delta_2: & \langle \sim h, t\rangle & \prec & \langle l_{\mathsf{S}}, t\rangle \\
\delta_3: & \langle h, t\rangle & \prec & \langle l_{\mathsf{N}}, t\rangle, \langle l_{\mathsf{S}}, t\rangle & \delta_4: & \langle l_{\mathsf{N}}, t\rangle & \prec & \langle \mu_{\mathsf{N}}, t\rangle \\
\delta_5: & \langle l_{\mathsf{S}}, t\rangle & \prec & \langle \mu_{\mathsf{S}}, t\rangle & \delta_6: & \langle b, t\rangle & \prec & \langle f, t\rangle \\
\delta_7: & \langle f, t+1\rangle & \prec & \langle \sim h, t\rangle, \langle o, t\rangle & \delta_8: & \langle \sim o, t\rangle & \prec & \langle f, t\rangle
\end{array}
$$

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

Assume that the two lifting actions are to be executed at the same time interval $[t-1, t]$, with $0 \leq t-1$ and $t \leq 10$, as in Figure 2.2 (Left). The arguments generated in this plan execution are the following.

$\mathcal{A}_1 = \{\delta_{\sim b}(t')\}_{0 \leq t' < 10} \cup \{\langle \sim b, 0\rangle\}$
$\mathcal{B}_1^- = \{\delta_1, \delta_4\} \cup \{\langle \mu_{\mathsf{N}}, t\rangle\}$
$\mathcal{B}_1 = \mathcal{B}_1^- \cup \{\delta_o(t')\}_{0 \leq t' < t} \cup \{\delta_6, \delta_7\}$
$\mathcal{A}_4 = \{\delta_{l_{\mathsf{N}}}(t')\}_{t \leq t' < 10} \cup \{\delta_4, \langle \mu_{\mathsf{N}}, t\rangle\}$

$\mathcal{A}_3 = \{\delta_3, \delta_4, \delta_5, \langle \mu_{\mathsf{N}}, t\rangle, \langle \mu_{\mathsf{S}}, t\rangle\}$
$\mathcal{B}_2^- = \{\delta_2, \delta_5\} \cup \{\langle \mu_{\mathsf{S}}, t\rangle\}$
$\mathcal{B}_2 = \mathcal{B}_2^- \cup \{\delta_o(t')\}_{0 \leq t' < t} \cup \{\delta_6, \delta_7\}$
$\mathcal{A}_5 = \{\delta_{l_{\mathsf{S}}}(t')\}_{t \leq t' < 10} \cup \{\delta_5, \langle \mu_{\mathsf{S}}, t\rangle\}$

2.4 A brief look at Forward Planning in t-DeLP

As with other planning frameworks that lie beyond classical planning, forward planning algorithms tend to be much simpler than their counterparts in backward planning. This is clearly the case of t-DeLP, where a plan in forward planning is simple set of actions, and the algorithm just requires the update function given by Definitions 2.3.4 and 2.3.7.

Definition 2.4.1 (Forward Plan. Solution)**.** Given a planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$ for some set of agents Ag, a *plan* for $\mathbb{M}$ is any subset A' which is non-overlapping for each agent in Ag. A plan $A' \subseteq A$ is a *solution* iff $G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A')$

Example 2.4.2. Following Example 2.3.9, we can illustrate how the solution *both agents lift the table at t* is generated in a stepwise fashion. This is simply the sequence of plans

$$\varnothing \quad \longmapsto \quad \{\mathsf{lift}.\mathsf{N}^{t-1}\} \quad \longmapsto \quad \{\mathsf{lift}.\mathsf{N}^{t-1}, \mathsf{lift}.\mathsf{S}^{t-1}\}$$

If the latter plan is generated by the planner, the BFS algorithm for forward search will check that the terminating condition $G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A')$ holds.

The proofs that a forward planning algorithm based on Breadth First Search is sound and complete is straightforward. Let us briefly present the search space and a BFS algorithm for it. For the search space,

(1) the set of nodes in the search space is $\mathcal{P}(A)$, so a state π is a set of actions $A' \subseteq A$. Each set of actions $A' \subseteq A$ is identified with the updated t-DeLP program $(\Pi, \Delta) \diamond A'$ –in accordance with the previous account of states as t-DeLP programs.

(2) the arcs in the search space are simply given by the (refinement) relation between pairs of sets of the form A' and $A' \cup \{\mathsf{e}\}$, for some $\mathsf{e} \notin A'$.

For simplicity, only the single-agent case is presented in Algorithm 1. For the multi-agent case, simply add the condition "*and* $\pi \cup \{\mathsf{e}\}$ *is non-overlapping in* Ag" to the set described in the line for Plans.

Data: $\mathbb{M} = ((\Pi, \Delta), A, G)$
Result: π; or fail
initialization: $\pi = \varnothing$ and $\mathsf{Plans} = \langle \pi \rangle$;
while $\mathsf{goals}(\pi) \not\subseteq \mathsf{warr}((\Pi, \Delta) \diamond \pi)$ **do**
  delete π from Plans;
  set $\mathsf{Plans} = \mathsf{Plans}^{\frown}\langle \pi \cup \{\mathsf{e}\} \mid \mathsf{e} \in A$ and $\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}((\Pi, \Delta) \diamond \pi)\rangle$;
  if $\mathsf{Plans} = \varnothing$ **then**
    set $\pi = \mathsf{fail}$
  else
    set $\pi =$ the first element of Plans
  end
end

Algorithm 1: Breadth First Search for t-DeLP forward planning.

What makes this algorithm an example of forward planning is the condition $\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}((\Pi, \Delta) \diamond \pi)$ requiring the new action e to be executable, according to the present plan π. In consequence, the effects of $\pi \cup \{\mathsf{e}\}$ will never be the same than those of π.

Theorem 2.4.3. *The BFS method of Algorithm 1 for t-DeLP forward planning is sound and complete.*

Proof. (Soundness) Let $A' \subseteq A$ be the output of the planning algorithm in Algorithm 1 for some $\mathbb{M} = ((\Pi, \Delta), A, G)$. That is, we set $A' = \pi$, the output for $\mathbb{M}$. On the one hand A' is non-overlapping for A', since this condition is satisfied by the last refinement step (each refinement step, in general). Observe that $G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A')$ since this is just the Terminating Condition, so A' is a solution.

(Completeness) Let $A' \subseteq A$ be a solution to $\mathbb{M} = ((\Pi, \Delta), A, G)$. Without loss of generality, we can assume that this solution A' is $\subseteq$-minimal. Let then $A' = \{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$. Let us assume that these actions are ordered in terms of increasing execution time; that is, $t_{\mathsf{e}_i} \leq t_{\mathsf{e}_{i+1}}$. Define $\pi_k = \{\mathsf{e}_1, \ldots, \mathsf{e}_k\}$. Since A' is a $\subseteq$-minimal solution, e_1 must be executable in (Π, Δ) (i.e. $\mathsf{pre}(\mathsf{e}_1) \subseteq \mathsf{warr}(\Pi, \Delta)$). Hence, $\{\mathsf{e}_1\}$ is a valid refinement of $\varnothing$. By an easy inductive proof, the same can be said w.r.t. each action e_{k+1} and plan π_k. Since, clearly, the set of refinements of each plan is finite (namely $|A|$) the plan $\pi = A'$ is eventually generated at most at turn $|A| + |A|^2 + \cdots + |A|^n$, and hence it will be the output of the algorithm, since being a solution it will satisfy the Terminating Condition. (All this, provided no other plan satisfying the Terminating Condition has already been found, in which case we would also be done.) □

Remark 2.4.4. Let us note that Algorithm 1 need not output $\subseteq$-minimal solutions. That is, the output π might properly contain a subset of actions $\pi' \subsetneq \pi$, such that π' is already a solution: $G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond \pi')$. The reason is that an action introduced earlier in the plan, when it was executable, stops being so after some other action is introduced. If the planner is interested in $\subseteq$- minimal

solutions, she can either: (1) introduce this notion of threat and prune those generated plans which contain a threat; or (2) she can add an additional requirement to Algorithm 1, namely that plans are refined in a time-increasing way, that is, for $\pi \cup \{\mathsf{e}\}$ to be a plan, it must also satisfy that $t_{\mathsf{f}} \leq t_{\mathsf{e}}$, for each $\mathsf{f} \in \pi$.

2.5 A t-DeLP planning system for backward search

The initial idea for t-DeLP backward planning is to start enforcing the goals as the conclusions of action-supported arguments and iteratively enforce the preconditions of those actions with more arguments. This will only work, though, if the undefeated status of these arguments is ultimately enforced as well. In the remaining of Part I, we will use *plan* to denote a plan built backwards, i.e. along these lines, in some planning domain.

In both forward- and backward-oriented planning systems, the notion of a (partial) state is given by a set of t-DeLP programs. A first difference between the two approaches, though, lies precisely in which states (t-DeLP programs) are actually considered by the planner. In the forward case, (partial) states are real possibilities, in the sense that executing the plan will lead to some "true" state (more or less close to the goal states). In the present backward approach, in contrast, all the actions makes sense w.r.t. the goals, but the planner does not know if these actions can ultimately be made executable (in further refinements of the plan). For this reason, we will refer to these states, and the dialectical trees they give rise to, as "provisional" states or trees. Another consequence of backward search, in t-DeLP planning, is the practical need to keep track of the dialectical trees for the planned arguments. (In comparison, forward plans were simply defined by their sets of actions). Although the topic of heuristic search for t-DeLP planning lies out of the present scope, it is noteworthy that this rich representation of backward plans might serve as well to study heuristics in t-DeLP planning. Thus, an estimation of the cost of solutions extending the current plan can be given in terms of the provisional dialectical trees for planned arguments.

Informally, a plan π for a given planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$ will be defined as a 3-tuple (actions, trees, goals) containing:

- a set of temporal actions $A(\pi) \subseteq A$,
- a set of dialectical trees $\mathsf{Trees}(\pi)$, one for each goal-enforcing argument, and
- a set of open goals $\mathsf{goals}(\pi) \subseteq \mathsf{TLit}$.

In practice, we will be only interested in those tuples (actions, trees, goals) that are generated by a node in the search space. In the search space, a plan (action, trees, goals) is identified with the node that generates it. Each nodes

in the search space is given by a sequence of plan steps (a succession of plan refinements). With more detail, the set of plans is the set of tuples that can be obtained from the empty plan (Def. 2.5.1) using a finite number of plan steps: either argument steps (Def. 2.5.3) to solve goals, or threat resolution moves (Def. 2.5.5) to defend arguments steps. For the sake of simplicity, we will assume that $G \cap \Pi_{\mathsf{f}} = \varnothing$. This permits to define next the open goals of the empty plan as the set of goals G, rather than the less elegant form $G \setminus \Pi_{\mathsf{f}}$.

Definition 2.5.1 (Empty plan)**.** The initial *empty plan* for a given planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$ is simply the triple

$$\pi_{\varnothing} = (\varnothing, \varnothing, G)$$

As we mentioned above, a plan can be specified as a sequence of n plan steps Λ_i (for $1 \leq i \leq n$), where each plan step Λ_i is either an argument step $\Lambda_i = [\mathcal{A}]$, or a threat resolution move $\Lambda_i = [\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}_i]$. Since such a sequence $\Lambda_1, \ldots, \Lambda_n$ is always applied to the empty plan, the resulting plan will be denoted

$$\pi_{\varnothing}(\Lambda_1, \ldots, \Lambda_i, \ \ldots, \Lambda_n)$$

If no confusion exists, this notation will be simplified by denoting the argument steps $\Lambda_i = [\mathcal{A}_i]$ or the threat resolution moves $\Lambda_i = [\mathcal{A}_j, \ldots, \mathcal{B}, \mathcal{A}_i]$ as follows:

$$\pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_i, \ldots, \mathcal{A}_n)$$

Recall that in backward classical planning, a plan is built by adding one plan step at a time (e.g an action), and the resulting plan simply replaces the goals enforced by this step (the effects) by new goals (the preconditions for the action to be executable). Analogously, if an argument $\mathcal{A}$ is used to enforce an open goal $\langle \ell, t \rangle$ in a plan π ($\langle \ell, t \rangle$ being the conclusion of $\mathcal{A}$), the set of open goals of the refined plan $\pi(\mathcal{A})$ will replace this $\langle \ell, t \rangle$ by the preconditions of actions supporting this argument. See Figure 2.3 (2) for an example of an argument step, addressing a previous goal in (1), and replacing it by new goals $\mathsf{pre}(\mathsf{e})$ in (2). That is, the set of literals $\mathsf{base}(\mathcal{A})$ consists of actions' effects and initial strict facts.

Such a rewriting of goals, though, does not suffice in general. We also need to make sure that the argument $\mathcal{A}$ will be undefeated in its own dialectical tree (so its conclusion is warranted). Thus, except for the case where $\mathcal{A}$ is a strict argument (so its undefeated status can be taken for granted), defeasible arguments $\mathcal{A}$ require the planner to maintain a list of (fragments of) provisional trees, denoted $\mathsf{Trees}(\pi)$. These trees keep track of existing defeaters for some such argument $\mathcal{A}$, enabled by actions already in the plan. These defeaters, called *threats*, are interfering arguments $\mathcal{B}$ in argumentation lines $[\mathcal{A}, \mathcal{B}]$ in the provisional tree for $\mathcal{A}$, see Fig. 2.3 (3). According to the t-DeLP marking procedure Def.1.3.22, the planner must defeat all of them, so at least a new defeater $\mathcal{C}$ for each threat $\mathcal{B}$ must be planned for, see Fig. 2.3 (4). (For practical reasons, this "at least" will turn rather into "exactly one" defeater $\mathcal{C}$ for each threat $\mathcal{B}$). At its turn,

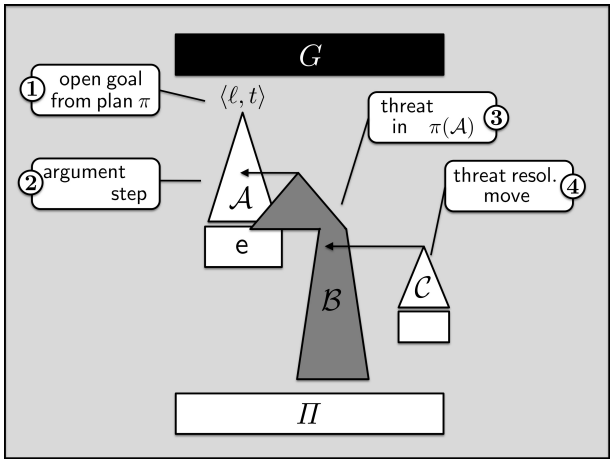


Figure 2.3: An argument step $\mathcal{A}$ introduces an action e which triggers an argument threat $\mathcal{B}$ to $\mathcal{A}$ itself. This threat is addressed by a further plan step $\mathcal{C}$.

these planned defeaters $\mathcal{C}$, called *threat resolution moves*, might be threatened by further arguments $\mathcal{D}$ in some argumentation line $[\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}]$, and so on.

To understand how these dialectical trees are computed during the plan construction, recall first that each set of actions $A' \subseteq A$ applied to a t-DeLP program (Π, Δ) generates a unique t-DeLP program $(\Pi, \Delta) \diamond A'$. This unique t-DeLP program will be referred to as the "true" state, i.e. the state that results after the execution of A' on (Π, Δ). In backward search, though, one initially abstracts from the executability of planned actions. Thus, rather than the "true" state, a t-DeLP backward planner has to work with a "provisional" state, a t-DeLP program generated by A' under the assumption that each action in A' will ultimately be executable (in some further plan refinement).

Definition 2.5.2 (Action Expansion)**.** For a given planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$, and a set of actions $A' \subseteq A$, we define the (provisional) *state* given by an *expansion with* A', denoted $(\Pi \oplus A', \Delta)$, as follows

$$(\Pi \oplus A', \Delta) = (\Pi \cup \mathsf{post}[A'], \Delta)$$

In particular, for a set of actions $A(\pi)$ of a plan π, we will use the notation $(\Pi \oplus \pi, \Delta) = (\Pi \oplus A(\pi), \Delta) = (\Pi \cup \mathsf{post}[A(\pi)], \Delta)$. In order to prevent this notation from becoming too cumbersome, we suggest the following notation for t-DeLP programs:

$\mathbb{P}$	denotes	(Π, Δ)	$\mathbb{P} \oplus A'$	denotes	$(\Pi \oplus A', \Delta)$
$\mathbb{P} \diamond \pi$	denotes	$(\Pi, \Delta) \diamond A(\pi)$	$\mathbb{P} \oplus \pi$	denotes	$(\Pi \oplus A(\pi), \Delta)$

In this notation, a set of planned actions $A(\pi)$ induces a "true" state $\mathbb{P} \diamond \pi$, and a "provisional" state $\mathbb{P} \oplus \pi$. Let us return to these provisional states. Each

provisional state $\mathbb{P} \oplus \pi$, being a t-DeLP program, will as usual generate a unique (full) dialectical tree $\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A})$ for each existing argument. The planner, though, will not be interested in the dialectical trees for planned arguments, but rather on an initial fragment, or sub-tree, of each such dialectical tree. Thus, the sub-tree of a provisional program considered by the planner, is denoted with a $*$ super-index $\mathcal{T}^{*}_{\mathbb{P}\oplus\pi}(\mathcal{A})$, and it will be called (again) a *provisional* tree (or *provisional* dialectical tree). In addition, the planner will not pay attention to arbitrary arguments that might exist, but only to those arguments $\mathcal{A}$ that have been planned for as argument steps in the plan.

As more and more actions are added to the plan, the provisional and full dialectical trees will grow accordingly: a maximal argumentation line $[\mathcal{A}_1, \ldots, \mathcal{A}_k]$ in a plan π ceases to be maximal, because in the refined plan π', an extended argumentation line $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{A}_{k+1}]$ exists, due to $\mathcal{A}_{k+1}$ being activated by the newer actions in π'. We will denote by $\sqsubseteq$ the relation *is an initial sub-tree of* , and also refer to this relation simply by *is a sub-tree of* . Using this notation, Figure 2.4 illustrates the fact that any provisional tree is a sub-tree of the full dialectical tree (for the same program) as well as a sub-tree of the provisional tree in a refined plan.

$$
\begin{array}{ccccccc}
\mathcal{T}^{*}_{\mathbb{P}\oplus\pi}(\mathcal{A}) & \sqsubseteq & \ldots & \sqsubseteq & \mathcal{T}^{*}_{\mathbb{P}\oplus\pi'}(\mathcal{A}) & & \\
\sqcap\!\mid & & \sqcap\!\mid & & \sqcap\!\mid & & \\
\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A}) & \sqsubseteq & \ldots & \sqsubseteq & \mathcal{T}_{\mathbb{P}\oplus\pi'}(\mathcal{A}) & = & \mathcal{T}_{\mathbb{P}\diamond\pi'}(\mathcal{A})
\end{array}
$$

Figure 2.4: An illustration of the sub-tree relation $\sqsubseteq$ among dialectical trees in the planning phase. The top line consists of *provisional* dialectical trees, while the bottom line consists of their *full* counterparts. The $\sqsubseteq$ relation holds between these pairs, and also between the dialectical trees of a plan and any of its plan refinements. The identity on the bottom-right corner states that the –planned for– undefeated status of $\mathcal{A}$ will actually be the case. If this holds for each argument step, this means that the plan π' is fully-executable; hence, if this π' is a solution, it is $\subseteq$-minimal.

A plan π, of course, need not always be refinable into a plan π', in which case π will be a terminal node in the search space. This will happen in particular when no goal-concluding or threat-resolving argument exists for any of the open goals or unsolved threats of the plan π. Four reasons can exist for this:

- first, no useful derivation exists: no combination of strict facts Π_{f}, rules $\Pi_{\mathsf{r}} \cup \Delta$ and action effects $\mathsf{post}[A]$ suffices to derive a goal $\langle \ell, t \rangle$, or to attack a threat;
- second, such derivations exist but they are not arguments: all of them are inconsistent with the strict fragment $\mathrm{Cn}(\Pi \oplus A(\pi))$;

- third (for threat-resolution only), some of these derivations are arguments $\mathcal{C}$ that attack a threat $\mathcal{B}$, but they do not generate an extended argumentation line $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}]$; this can be due to $\mathcal{C}$ not being a (proper) defeater for $\mathcal{B}$, or any other condition from Def. 1.3.18 not being satisfied; and

- fourth, that the actions required violate the non-overlapping condition.

The open goals after a refinement with an argument step for some goal are defined analogously as in temporal planning (the old plan's goals minus the conclusion plus the preconditions of the actions supporting the new plan step), and similarly for threat resolution moves. In particular, if a new plan step requires a goal already solved by a previous argument step, this latter demand for the same goal can be safely ignored.

In the following we will denote an initial segment of a plan, $\pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ up to its k-th element as π_k, i.e. $\pi_k = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_k)$. In particular π_0 denotes the empty plan $\pi_\varnothing$. Also notice that we forbid argument steps $\mathcal{A}$ to directly support (the base of) other argument steps $\mathcal{A}'$, already in the plan. Instead, the planner will have to consider a single argument step $\mathcal{A} \cup \mathcal{A}'$. This is another way of saying that the planned arguments must be fully based upon action effects and strict facts. [2]

Definition 2.5.3 (Argument Step Refinement)**.** Let $\mathbb{M} = (\mathbb{P}, A, G)$ be a planning domain for some set of agents Ag, where $\mathbb{P} = (\Pi_\mathsf{f} \cup \Pi_\mathsf{r}, \Delta)$. Let $\pi = \pi_n = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a plan for $\mathbb{M}$. Define first

$$\mathsf{OldGoals}(\pi_n) = \bigcup_{1 \leq k \leq n} (\mathsf{goals}(\pi_k) \setminus \bigcup_{k < k' \leq n} \mathsf{goals}(\pi_{k'}))$$

Then, let $\langle \ell, t \rangle \in \mathsf{goals}(\pi)$ be an open goal in π. Let $A^* \subseteq A$ and $\mathcal{A}^- \subseteq \Pi_\mathsf{r} \cup \Delta$ be sets of actions and rules satisfying:

(i) $A(\pi) \cup A^*$ is non-overlapping

(ii) $(\mathbb{P} \oplus \pi) \oplus A^*$ is a t-DeLP program

(iii) $\mathcal{A} = \mathcal{A}^- \cup \mathsf{base}(\mathcal{A}^-)$ is an argument for $\langle \ell, t \rangle$ in this program $(\mathbb{P} \oplus \pi) \oplus A^*$ (so $\mathsf{base}(\mathcal{A}) = \mathsf{base}(\mathcal{A}^-) \subseteq \Pi_\mathsf{f} \cup \mathsf{post}[A(\pi) \cup A^*]$)

(iv) A^* is $\subseteq$-minimal with the last property

(v) $\mathsf{pre}[A^*] \cup \mathsf{literals}(\mathcal{A}) \cup \bigcup_{\mathcal{A}_k \text{ arg. step in } \pi_n} \mathsf{literals}(\mathcal{A}_k) \cup \bigcup_{0 \leq k \leq n} \mathsf{goals}(\pi_k)$ is consistent

Then we denote the *refinement* of π *by* $\mathcal{A}$, as the new plan $\pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A})$, also denoted $\pi(\mathcal{A})$, defined by the components:

[2] Recall that $\mathsf{base}(\mathcal{A})$ is generally defined $\mathsf{base}(\mathcal{A}) = \mathsf{body}[\mathcal{A}] \setminus \mathsf{head}[\mathcal{A} \setminus \Pi_\mathsf{f}]$. In the following, we extend this definition to arbitrary sets of rules and facts, not only arguments.

$$
\begin{array}{rcl}
A(\pi(\mathcal{A})) & = & A(\pi) \cup A^* \\
\mathsf{goals}(\pi(\mathcal{A})) & = & (\mathsf{goals}(\pi) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi_{\mathsf{f}} \cup \mathsf{OldGoals}(\pi)) \\
\mathsf{Trees}(\pi(\mathcal{A})) & = & \{\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A}_k) : 1 \leq k \leq n\} \cup \{\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A})\} \\
 & & \text{where these trees are defined as follows} \\
\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A}_k) & = & \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}_k) \cup \{\Lambda^\frown[\mathcal{B}] \in \mathcal{T}_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A}_k) \mid \Lambda \text{ is a plan step in } \pi\} \\
\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A}) & = & \{[\mathcal{A}]\} \cup \{[\mathcal{A}, \mathcal{B}] \mid [\mathcal{A}, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}\oplus\pi(\mathcal{A})}(\mathcal{A})\}
\end{array}
$$

Each maximal argumentation line $\Lambda \in \mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{A})}(\cdot) \in \mathsf{Trees}(\pi(\mathcal{A}))$ with an even number of elements is called a *threat in* $\pi(\mathcal{A})$, and denoted $\Lambda \in \mathsf{threats}(\pi(\mathcal{A}))$.

Notice that a threat Λ in a sub-tree need not be a maximal argumentation line in the corresponding full dialectical tree. For example, in Fig. 2.5, the threat $[\mathcal{A}_1, \mathcal{D}_2]$ is not maximal in $\mathcal{T}_{\mathbb{P}\oplus\pi(\cdot)}(\mathcal{A}_1)$, since $[\mathcal{A}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4]$ exists in this tree.

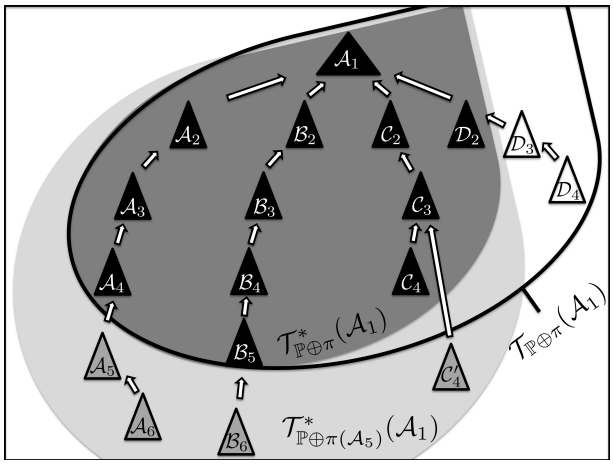


Figure 2.5: An illustration of the dialectical trees involved in in the refinement of a plan π with an argument step $\mathcal{A}_5$. The dark area represents the provisional tree for an argument step $\mathcal{A}_1$ in the plan π. Some defending arguments exist, e.g. $\mathcal{D}_3$, which are not planned for as threat resolution moves, and hence not part of this tree $\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A}_1)$. After a refining π with $\mathcal{A}_5$, new threats are generated, namely $\mathcal{A}_6, \mathcal{B}_6, \mathcal{C}'_4$.

Example 2.5.4. See Figure 2.5 for an illustration of the sub-tree $\mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}_1) \in \mathsf{Trees}(\pi)$ that corresponds to some plan π. This tree consists of part of the full tree $\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A}_1)$: some argumentation lines, e.g. $[\mathcal{A}_1, \mathcal{D}_2]$, are considered; while others, e.g. $[\mathcal{A}_1, \mathcal{D}_2, \mathcal{D}_3]$, are not. This means that $\mathcal{D}_3$ could still be used as a plan step against the threat $[\mathcal{A}_1, \mathcal{D}_2]$ for free, since it will be activated anyway by the plan. Whether it is used or not depends on its being explicitly considered as a threat resolution moves in in further refinements.

When a new step, e.g. the threat resolution move $\mathcal{A}_5$ is considered, the provisional trees in $\mathsf{Trees}(\pi)$ must be updated accordingly into $\mathsf{Trees}(\pi(\mathcal{A}_5))$. For instance, $\mathcal{A}_5$ will bring about the new argumentation line $[\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_5]$. But when we expand the old actions in π with the new actions required to support $\mathsf{base}(\mathcal{A}_5)$, the joint effects can trigger new threats: to $\mathcal{A}_5$ itself, e.g. $[\mathcal{A}_1, \ldots, \mathcal{A}_5, \mathcal{A}_6]$; or to other defending arguments in this sub-tree, e.g. $\mathcal{C}'_4$ as a threat to $[\mathcal{A}_1, \mathcal{C}_2, \mathcal{C}_3]$, or $\mathcal{B}_6$ as a threat to $[\mathcal{A}_1, \mathcal{B}_2, \ldots, \mathcal{B}_5]$. Finally, the new actions incorporated in the plan refinement with $\mathcal{A}_5$ can even bring about a new threat to the provisional tree of other argument steps ($\neq \mathcal{A}_1$) in the plan.

As we said, a planner will deal with a threat $[\mathcal{A}_1, \ldots, \mathcal{A}_{2n+2}]$ by planning for some defeater $\mathcal{C}$ for $\mathcal{A}_{2n+2}$. The joint effort of all these threat resolution moves in the provisional tree $\mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}_1)$ must ultimately grant that the argument step $\mathcal{A}_1$ is undefeated. Note that the argument $\mathcal{C}$ can occur several times in a plan, e.g. abusing notation we can have $\pi = \pi_\varnothing(\ldots, \mathcal{C}, \ldots, \mathcal{C}, \ldots)$. This argument $\mathcal{C}$ can occur only once as an argument step, and several times as different threat resolution moves.

Definition 2.5.5 (Threat resolution)**.** Let $\pi = \pi_n = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be a plan for $\mathtt{M} = (\mathbb{P}, A, G)$, with $\mathbb{P} = (\Pi_\mathsf{f} \cup \Pi_\mathsf{r}, \Delta)$. And let $\Lambda = [\mathcal{A}, \ldots, \mathcal{B}]$ be a threat $\Lambda \in \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}) \in \mathsf{Trees}(\pi)$ for some argument step $\mathcal{A} = \mathcal{A}_j$ with $1 \leq j \leq n$. Finally, let $A^* \subseteq A$ and $\mathcal{C}^- \subseteq \Pi_\mathsf{r} \cup \Delta$ be sets satisfying:

(i) $A(\pi) \cup A^*$ is non-overlapping

(ii) $(\mathbb{P} \oplus \pi) \oplus A^*$ is a t-DeLP program

(iii) $\mathcal{C} = \mathcal{C}^- \cup \mathsf{base}(\mathcal{C}^-)$ is an argument in $(\mathbb{P} \oplus \pi) \oplus A^*$ and, moreover, $\Lambda^\cap[\mathcal{C}]$ is an argumentation line for $\mathcal{A}$ in $(\mathbb{P} \oplus \pi) \oplus A^*$

(iv) A^* is $\subseteq$-minimal with the last property

Then we say that $\pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n, \Lambda^\cap[\mathcal{C}])$, also be denoted $\pi(\Lambda^\cap[\mathcal{C}])$ or simply $\pi(\mathcal{C})$, is the *refinement of* π *by* $\mathcal{C}$, defined by the components

$$
\begin{array}{rcl}
A(\pi(\mathcal{C})) & = & A(\pi) \cup A^*, \\
\mathsf{goals}(\pi(\mathcal{C})) & = & (\mathsf{goals}(\pi) \cup \mathsf{pre}[A^*]) \smallsetminus (\Pi_\mathsf{f} \cup \mathsf{OldGoals}(\pi)) \\
\mathsf{Trees}(\pi(\mathcal{C})) & = & \{\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{C})}(\mathcal{A}_k) \mid 1 \leq k \leq n\} \\
 & & \text{where each } \mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{C})}(\mathcal{A}_k) \text{ is defined as follows} \\
\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{C})}(\mathcal{A}_i) & = & \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}_i) \cup \\
 & & \{\Lambda'^\cap[\mathcal{B}'] \in \mathcal{T}_{\mathbb{P}\oplus\pi(\mathcal{C})}(\mathcal{A}_i) \mid \Lambda' \text{ is a plan step in } \pi\}, \text{ and} \\
\mathcal{T}^*_{\mathbb{P}\oplus\pi(\mathcal{C})}(\mathcal{A}) & = & \mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}) \cup \\
 & & \{\Lambda^\cap[\mathcal{C}]\} \cup \{\Lambda^\cap[\mathcal{C}, \mathcal{B}'] \mid \Lambda^\cap[\mathcal{C}, \mathcal{B}'] \in \mathcal{T}_{\mathbb{P}\oplus\pi(\mathcal{C})}(\mathcal{A})\}
\end{array}
$$

The threats of $\pi(\mathcal{C})$ are defined as in Def. 2.5.3.

Definitions 2.5.3 and 2.5.5 describe all the possible ways a plan π can be refined. Indeed, the *space of plans* for $\mathbb{M}$ is precisely the set of plans that can be obtained by a finite number of applications of these refinements upon the empty plan $\pi_\varnothing$ for $\mathbb{M}$.

Definition 2.5.6 (Plan. Solution)**.** Given a planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$, we say π is a *plan* for $\mathbb{M}$ iff it is either the empty plan $\pi_\varnothing = (\varnothing, \varnothing, G)$, or it obtains from $\pi_\varnothing$ after a finite number of applications of Definitions 2.5.3 and 2.5.5. We say a plan π is a *solution* for $\mathbb{M}$ iff $G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond \pi)$.

Also notice that the order in which actions are added to the plan need not be the reverse of the execution ordering for these actions. For example, a new threat resolution move might require to schedule a new action between two actions already in the plan.

Let us conclude this section by describing Example 2.3.9 in full detail.

Example 2.5.7 (Table Lifting; cont'd)**.** Following Example 2.3.9, we illustrate a planning task with a sequence of plan steps which lead to a solution plan. The central planner can find that the plan *both agents lift the table at t* is a solution by the following sequence of plan refinements:

$\pi_\varnothing$	open goals G; no threats
$\pi_\varnothing(\mathcal{A}_1)$	solves goal $\langle \sim b, 10\rangle$; adds no goals
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4)$	solves $\langle l_{\mathsf{S}}, 10\rangle$; new threat $[\mathcal{A}_1, \mathcal{B}_1]$
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4, \mathcal{A}_5)$	solves $\langle l_{\mathsf{N}}, 10\rangle$; new threat $[\mathcal{A}_1, \mathcal{B}_2]$
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4, \mathcal{A}_5, [\mathcal{A}_1, \mathcal{B}_1, \mathcal{A}_3])$	solves $[\mathcal{A}_1, \mathcal{B}_1]$
$\pi_\varnothing(\mathcal{A}_1, \mathcal{A}_4, \mathcal{A}_5, [\mathcal{A}_1, \mathcal{B}_1, \mathcal{A}_3], [\mathcal{A}_1, \mathcal{B}_2, \mathcal{A}_3])$	solves $[\mathcal{A}_1, \mathcal{B}_2]$; this is a solution

Figure 2.2(right), in contrast, contains a threat $\mathcal{B}_1$ which cannot be solved once it occurs in plans based on a non-simultaneous lifting of the two sides of the table. These plans do not end up in solution plans.

In summary, the (action-based) arguments make $\langle \sim b, t\rangle$, $\langle l_{\mathsf{N}}, t\rangle$ and $\langle l_{\mathsf{S}}, t\rangle$ warranted iff agents lift both sides simultaneously, see Fig. 2.2 (Left).

This concludes the description of the planning system. In the remaining of the chapter, we introduce and study several search algorithms for the present space of plans.

2.6 Algorithms for t-DeLP backward planning

We will consider the *space of plans* for a planning domain $\mathbb{M}$, defined as the graph given by the following sets of nodes and arcs:

- (nodes) the set of plans π that can be obtained by a finite number of applications of Defs. 2.5.3 and 2.5.5, starting from the root node $\pi_\varnothing$ and
- (arcs) the relation π' *is a refinement of* π, i.e. the set of pairs (π, π') satisfying this condition.

Let us present the Breadth First Search algorithm for multi-agent t-DeLP planning domains.

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

Algorithm 2: Breadth First Search for backward planning in the t-DeLP planning system.

Since G is a finite set of temporal literals $\langle \ell, t \rangle$, these goals are bounded by some maximum value t^*, and so plan steps simply consist of arguments (and arguments) whose conclusions (resp. effects) are about some $t \leq t^*$. This implies that there are only finitely-many plan step refinements of any plan, that can be obtained from Δ and A. In other words, not only the space of plans is a finitely branching tree (since $\mathcal{P}(\Delta)$ and A are finite), but moreover this space is finite.

As a consequence, the usual search methods BFS, DFS, etc are terminating, so the following proofs for BFS can be easily adapted to other well-known search algorithms besides BFS.

For example, for DFS we would simply order newly generated plans before the plans generated previously, so Algorithm 2 can be adapted as follows for DFS:

replace	set $\mathsf{Plans} = \mathsf{Plans} \cap \langle\, \pi(\mathcal{A}) \mid \pi(\mathcal{A})$ is a refinement of $\pi \,\rangle$
by	set $\mathsf{Plans} = \langle\, \pi(\mathcal{A}) \mid \pi(\mathcal{A})$ is a refinement of $\pi \,\rangle \cap \mathsf{Plans}$

2.7 Soundness of BFS search for backward t-DeLP planning

In this section, we proceed to show that the BFS algorithm for the t-DeLP backward planning system is sound: so if this algorithm outputs some plan π for a planning domain $\mathbb{M}$ given as input, then π is a solution for $\mathbb{M}$. A first instrumental result for this says that the existing arguments are preserved under plan refinements.

Lemma 2.7.1. *Let $\mathbb{M} = (\mathbb{P}, A, G)$ be a planning domain, π a plan for $\mathbb{M}$ and $\pi(\mathcal{A})$ a plan refinement of π. Then, for any argument $\mathcal{A}'$ in $\mathbb{P} \oplus \pi$, we have that*

$\mathcal{A}'$ *is also an argument in* $\mathbb{P} \oplus \pi(\mathcal{A})$.

Proof. Let $\mathbb{P} = (\Pi, \Delta)$ as usual with $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathsf{r}}$. We check the conditions (1)-(4) from Def. 1.3.5.

(1) Since we have $\Pi \oplus \pi_k \subseteq \Pi \oplus \pi_{k+1}$, all the derivations from $\mathbb{P} \oplus \pi_k$ exist as well in $\mathbb{P} \oplus \pi_{k+1}$.

(2) Assume towards a contradiction that $\Pi \oplus \pi(\mathcal{A})$ is inconsistent with $\mathcal{A}' \cap \Delta$. Say $\langle \ell, t\rangle, \langle \sim\ell, t\rangle$ are both derivable from all these facts and rules. Since $\Pi \oplus \pi$ is consistent with $\mathcal{A}'$, one of these two literals, say $\langle \sim\ell, t\rangle$, is not derivable from $(\Pi \oplus \pi) \cup \mathcal{A}'_\Delta$. Thus, the derivation for $\langle \sim\ell, t\rangle$ in $\mathbb{P} \oplus \pi(\mathcal{A})$ must make use of the new information in the latter set of strict facts and rules. Without loss of generality, assume that the derivation for $\langle \sim\ell, t\rangle$ is $\subseteq$-minimal. The previous claim on the need for some new strict information, together with the fact that

$$\mathbb{P} \oplus \pi(\mathcal{A}) = (\mathbb{P} \oplus \pi) \oplus A^*, \quad \text{for some set of actions } A^*$$

implies that the derivation for $\langle \sim\ell, t\rangle$ contains some literal $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$ as a premise. Since $\langle \sim\ell, t\rangle \neq \langle \sim\mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle \neq \langle \ell, t\rangle$ (by def. of μ_{e}, its negation cannot occur in a program), the $\subseteq$-minimality of this derivation implies that some rule δ exists in this derivation with $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle \in \mathsf{body}(\delta)$. We show this is impossible. (Case $\delta \in \Delta$) That is, $\delta \in \mathcal{A}' \cap \Delta$. Since $\mathcal{A}'$ is a $\subseteq$-minimal argument in $\mathbb{P} \oplus \pi_k$, this implies that $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle \in \mathbb{P} \oplus \pi_k$, contradicting the assumption on $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$. (Case $\delta \in \Pi_{\mathsf{r}}$) Since $\Pi_{\mathsf{r}} = \Pi_{\mathbf{M}}$, this rule δ must be a mutex rule, but this contradicts the assumptions on the μ_{e}-effects.

(3) First, assume towards a contradiction that $\mathcal{A}' \cap \Delta$ is not $\subseteq$-minimal w.r.t. condition (1). Then some $\subseteq$-minimal $\mathcal{A}''_\Delta \subsetneq \mathcal{A}'_\Delta$ exists with $\mathcal{A}''_\Delta \cup \mathbb{P} \oplus \pi_{k+1} \vdash \mathsf{concl}(\mathcal{A})$. Then, as before we can reason that, since $\mathcal{A}'$ is an argument in $\mathbb{P} \oplus \pi_k$, we must have some $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle \in \mathsf{base}(\mathcal{A}'')$. And from the minimality of $\mathcal{A}''$, some rule δ exists with $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$. In either case $\delta \in \Pi_{\mathsf{r}}$ or $\delta \in \Delta$ we reach a similar contradiction than we did in (2). Thus, $\mathcal{A}' \cap \Delta$ is $\subseteq$-minimal w.r.t. (1). On the other hand, it is obvious, given (2), that $\mathcal{A}' \cap \Delta$ is $\subseteq$-minimal w.r.t. (2) also in $\mathbb{P} \oplus \pi_{k+1}$.

(4) Assume towards a contradiction, that some $\subseteq$-mimimal $\mathcal{A}''_\Pi \subsetneq \mathcal{A}'_\Pi$ exists with the property $\mathcal{A}''_\Pi \cup \mathcal{A}'_\Delta \vdash \mathsf{concl}(\mathcal{A}')$. But this is impossible, since then $\mathcal{A}''_\Pi \cup \mathcal{A}'_\Delta \subseteq \mathcal{A}'$ implies that $\mathcal{A}''_\Pi \cup \mathcal{A}'_\Delta$ exists in $\mathbb{P} \oplus \pi_k$, so $\mathcal{A}'$ does not satisfy condition (4) in $\mathbb{P} \oplus \pi_k$. □

Lemma 2.7.2. *Let* $\mathbb{M}$ *be a planning domain and* π_k *some plan for* $\mathbb{M}$. *If* π_{k+1} *is a refinement of* π, *then, for each argumentat step* $\mathcal{A}$ *in* π_k, *we have that*

$$\mathcal{T}_{\mathbb{P}\oplus\pi_k}(\mathcal{A}) \sqsubseteq \mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$$

Proof. For this, assume the contrary towards a contradiction. That is, assume that some argumentation line $[\mathcal{A}, \ldots, \mathcal{A}'] \in \mathcal{T}_{\mathbb{P}\oplus\pi_k}(\mathcal{A})$ is not an argumentation line in $\mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$. We check each case where some condition from Def. 1.3.18

might fail. By Lemma 2.7.1, all these $\mathcal{A}, \ldots, \mathcal{A}'$ are arguments in $\mathbb{P} \oplus \pi_{k+1}$ as well, and each of them is a defeater for its predecessor in Λ. Thus, it only remains to consider a failure of some condition (i)-(iii) from Def. 1.3.18.

(i) Assume condition (i) fails, e.g. $\Pi \oplus \pi_{k+1}$ is inconsistent with the defending arguments of Λ. Then, let $\langle \ell, t\rangle, \langle \sim\ell, t\rangle$ be the inconsistent pair derivable from Π plus the rules used in defending arguments. By assumption, this inconsistency does not exist in $\mathbb{P} \oplus \pi_k$, so one of the two literals, say $\langle \sim\ell, t\rangle$, is not derivable from this strict information and the rules from the defending arguments. Thus, a derivation for it using $\mathbb{P} \oplus \pi_{k+1}$ must make use of some new strict fact in $\mathsf{post}[A^*]$ where $A(\pi_{k+1}) = A(\pi_k) \cup A^*$. Assume this derivation is $\subseteq$-minimal. Say $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$ is a premise in this derivation for $\langle \sim\ell, t\rangle$, for some $\mathsf{e} \in A^*$. By the $\subseteq$-minimality of A^* (condition (iv) from Def. 2.5.3), $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$ cannot occur in the defending arguments, all based on $\mathbb{P} \oplus \pi_k$. Thus, some rule δ is needed in the derivation with the property $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle \in \mathsf{body}(\delta)$. (Case $\delta \in \Delta$) This is impossible, again because this would imply that δ occurs in some defending argument $\mathcal{A}''$, so by the $\subseteq$-minimality of the argument $\mathcal{A}''$, it contains $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle \in \mathsf{base}(\mathcal{A}'') \subseteq \Pi \oplus \pi_k$. This would violate the $\subseteq$-minimality of A^*. (Case $\delta \in \Pi_{\mathsf{r}}$) This is impossible, since by definition no effect $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$ occurs in a mutex rule in $\Pi_{\mathbf{M}}$, and the assumption on mutex programs $\Pi_{\mathbf{M}} = \Pi_{\mathsf{r}}$. This shows condition (i) for the defending arguments. The proof for the consistency of the interfering arguments with $\Pi \oplus \pi_{k+1}$ is analogous.

(ii) Clearly, condition (ii) holds because the attacked sub-arguments in Λ are the same in either t-DeLP program $\mathbb{P} \oplus \pi_k$ and $\mathbb{P} \oplus \pi_{k+1}$.

(iii) The arguments in the sequence $[\mathcal{A}, \ldots, \mathcal{A}']$ stands in the same $\succ$ or $\prec\!\!\succ$ defeat relations (w.r.t their predecessors) in either program $\mathbb{P} \oplus \pi_k$ and π_{k+1}, so condition (iii) is satisfied. □

Another result concerns the (provisional) sub-trees used during the plan construction. Each of these sub-trees is equivalent, in terms of the marking procedure of Def. 1.3.22, to the full dialectical tree for the same argument. As a consequence, the undefeated status of arguments and the warrant status of their conclusions (goals) is the same between these trees.

Lemma 2.7.3. *Let $\mathbb{P} = (\Pi, \Delta)$ be an arbitrary t-DeLP program, and let $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$ be the dialectical tree for some argument $\mathcal{A}_1$ in $\mathbb{P}$. Finally, let $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$ be an arbitrary sub-tree of $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$ with the following properties:*

(i) the argumentation line $[\mathcal{A}_1]$ is in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$

(ii) for any an arg. line $\Lambda = [\mathcal{A}_1, \ldots, \mathcal{A}_{2n}]$ in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$, if some $\Lambda^+ = [\mathcal{A}_1, \ldots, \mathcal{A}_{2n}, \mathcal{A}_{2n+1}]$ exists in $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$ with $\mathcal{A}_{2n+1}$ undefeated then Λ^+ is also in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$ for a unique such Λ^+

(iii) for any arg. line $\Lambda = [\mathcal{A}_1, \ldots, \mathcal{A}_{2n+1}]$ in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$ and any $\Lambda^+ = [\mathcal{A}_1, \ldots, \mathcal{A}_{2n+1}, \mathcal{A}_{2n+2}]$ in $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$ we have that Λ^+ is also in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$

Then, for any argumentation line $[\mathcal{A}_1, \ldots, \mathcal{A}_k]$ *in* $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$,

$$\mathcal{A}_k \textit{ is undefeated in } \mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1) \quad \textit{iff} \quad \mathcal{A}_k \textit{ is undefeated in } \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$$

Moreover, if in such a sub-tree $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$*, each interfering argument has a defeater, then* $\mathcal{A}_1$ *is undefeated in this tree. The same holds for* $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$*.*

Proof. The proof of the first claim is by induction on the (sub-)tree structure.

(Base Case $\mathcal{A}_k$ is terminal in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$.) By the marking procedure Def. 1.3.22, we have on the one hand that $\mathcal{A}_k$ is undefeated in $[\mathcal{A}_1, \ldots, \mathcal{A}_k] \in \mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$. To show this $\mathcal{A}_k$ is also undefeated in $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$, consider the two cases. (Case k is odd.) Then, applying condition (iii) from the Lemma we conclude that no defeater $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$, so $\mathcal{A}_k$ is terminal in this tree as well, and hence it is undefeated. (Case k is even.) Then, by condition (ii), no defeater for $\mathcal{A}_k$ in $[\mathcal{A}_1, \ldots, \mathcal{A}_k]$ exists in $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$. Again, this argument $\mathcal{A}_k$ is terminal in $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$ and so undefeated in this tree.

(Inductive Case) Assume (Ind. Hyp.) that the equivalence in the undefeated status holds for each argument $\mathcal{B}$ in any argumentation line of the form $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \ldots, \mathcal{B}, \ldots] \in \mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$. We have that

$$\begin{aligned} & \mathcal{A}_k \text{ is undefeated in } \mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1) \\ \text{iff} \quad & \text{for each } [\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}] \in \mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1), \quad \mathcal{B} \text{ is defeated} \\ \text{iff} \quad & \text{for each } [\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1), \quad \mathcal{B} \text{ is defeated} \end{aligned}$$

We proceed to show the latter equivalence. The ($\Uparrow$) direction follows from the fact that $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1) \sqsubseteq \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$ and the Ind. Hyp. For the ($\Downarrow$) direction, let $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}]$ be arbitrary in $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$. Consider the two cases. (Case k is odd.) Then, by condition (iii), $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}]$ is also in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$, and by the Ind. Hyp. $\mathcal{B}$ must also be defeated in $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$. (Case k is even.) If $\mathcal{B}$ was undefeated in $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$, by condition (ii) some $\mathcal{B}'$ also undefeated in $[\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}]$ would exist in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$. By the Ind. Hyp., this $\mathcal{B}$ would be undefeated in $\mathcal{T}^*_{\mathbb{P}}(\mathcal{A}_1)$, contradicting the assumption.

Finally, we can resume the above line of reasoning as follows

$$\begin{aligned} & \text{for each } [\mathcal{A}_1, \ldots, \mathcal{A}_k, \mathcal{B}] \in \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1), \quad \mathcal{B} \text{ is defeated} \\ \text{iff} \quad & \mathcal{A}_k \text{ is undefeated in } \mathcal{T}_{\mathbb{P}}(\mathcal{A}_1) \end{aligned}$$

which concludes the proof of the Ind. Case.

For the second claim, assume towards a contradiction that each interfering argument (in any arg. line) has a defeater but that $\mathcal{A}_1$ is defeated. The latter implies the existence of some defeater $\mathcal{A}_2$ which is undefeated. By the former assumption, some defeater $\mathcal{A}_3$ for $\mathcal{A}_2$ must exist, which is defeated. This reasoning can be repeated indefinitely, so as to give argumentation lines $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots, \mathcal{A}_{2k}, \mathcal{A}_{2k+1}]$ of arbitrary length k. But this contradicts the fact that the dialectical tree $\mathcal{T}_{\mathbb{P}}(\mathcal{A}_1)$ is finite, so we are done. □

Before proceeding with the soundness theorem, we prove that the previous Lemma 2.7.3 can be applied if no threats exist in a plan π. That is, we show

that in this case the trees $\mathcal{T}^*_{\mathbb{P}\diamond\pi}(\cdot)$ satisfy the conditions in the previous Lemma 2.7.3.

Lemma 2.7.4. *Let $\mathbb{M}$ be a planning domain and $\pi = \pi_n = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ an arbitrary plan for $\mathbb{M}$ with the property $\mathsf{threats}(\pi) = \varnothing$. Then for any argument step $\mathcal{A}$ in π, the tree $\mathcal{T}^*_{(\mathbb{P}\oplus\pi)}(\mathcal{A})$ (computed using by Definitions 2.5.1, 2.5.3 and 2.5.5) is a subtree of $\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A})$ that satisfies conditions (i)-(iii) from Lemma 2.7.3.*

Proof. First, we prove by induction that any tree $\mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A})$ obtained using Definitions 2.5.1, 2.5.3 and 2.5.5 is a sub-tree of $\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A})$.

(Base Case $\pi_\varnothing$) This is obvious, since no argument step exists in $\pi_\varnothing$.

(Ind. Case $\pi_k \Rightarrow \pi_{k+1}$.) Assume (Ind. Hyp.), that $\mathcal{T}^*_{\mathbb{P}\oplus\pi_k}(\mathcal{A}) \sqsubseteq \mathcal{T}_{\mathbb{P}\oplus\pi_k}(\mathcal{A})$ for any argument step $\mathcal{A}$. We consider the next two cases.

(Case $\mathcal{A}_{k+1}$ is an argument step.) First consider the case $\mathcal{A} = \mathcal{A}_{k+1}$. Since $\mathcal{A}_{k+1}$ is an argument in $\mathbb{P} \oplus \pi_{k+1}$, Definition 2.5.3 gives that $\mathcal{T}^*_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A}_{k+1})$ is defined as a sub-tree of $\mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A}_{k+1})$. Second, for any other argument step $\mathcal{A} \neq \mathcal{A}_{k+1}$, Def. 2.5.3 defines

$$\mathcal{T}^*_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A}) = \mathcal{T}^*_{\mathbb{P}\oplus\pi_k}(\mathcal{A}) \cup \{\Lambda^\frown[\mathcal{B}] \in \mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A}) \mid \Lambda \text{ is a plan step in } \pi_k\}$$

Now, since *being a plan step* Λ *in plan* π_k implies $\Lambda \in \mathcal{T}^*_{\mathbb{P}\oplus\pi_k}(\mathcal{A})$ (by Def. 2.5.3 or Def. 2.5.5), the latter set in the union is a set of paths in $\mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$. At its turn, the former set in the union satisfies $\mathcal{T}^*_{\mathbb{P}\oplus\pi_k}(\mathcal{A}) \sqsubseteq \mathcal{T}_{\mathbb{P}\oplus\pi_k}(\mathcal{A})$ (by the Ind. Hyp.); moreover, by Lemma 2.7.2 we have that $\mathcal{T}_{\mathbb{P}\oplus\pi_k}(\mathcal{A}) \sqsubseteq \mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$, so combining the last two $\sqsubseteq$-claims, we conclude that the former set in the union is also a sub-tree of $\mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$. Thus, all the paths in the above union are also in $\mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$. This and the obvious fact that this union is closed under initial fragments of its paths (argumentation lines) jointly imply that the above union is a sub-tree of $\mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$. If we replace this union by the set it defines, we obtain $\mathcal{T}^*_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A}) \sqsubseteq \mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$, as desired.

(Case $\mathcal{A}_{k+1}$ is a threat resolution move $[\ldots, \mathcal{A}_{k+1}]$) The proof for this case is analogous, using Definition 2.5.5 instead of Definition2.5.3.

This concludes the proof for $\mathcal{T}^*_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A}) \sqsubseteq \mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}}(\mathcal{A})$.

We proceed to show conditions (i)-(iii) from Lemma 2.7.3 for the t-DeLP program $\mathbb{P} \oplus \pi$ induced by π, and arbitrary argument steps $\mathcal{A}$. Let then $\mathcal{A}$ be an argument step in π.

(i) Since $\mathcal{A}$ is an argument step, say $\mathcal{A} = \mathcal{A}_k$, then $[\mathcal{A}] \in \mathcal{T}^*_{\mathbb{P}\oplus\pi_k}(\mathcal{A})$. A look into Definitions 2.5.3 and 2.5.5 (for $\pi_{k+1}, \ldots, \pi_n$) should suffice to convince oneself that $[\mathcal{A}]$ is also in the sub-tree $\mathcal{T}^*_{\mathbb{P}\oplus(\cdot)}(\mathcal{A})$ corresponding to each of these plans.

(ii) Let $\Lambda = [\mathcal{A}, \ldots, \mathcal{B}]$ be an even-length argumentation line in $\mathcal{T}^*_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$. Let $k \leq n$ be minimal with the property $[\mathcal{A}, \ldots, \mathcal{B}] \in \mathcal{T}^*_{\mathbb{P}\oplus\pi_k}(\mathcal{A})$. Note that we cannot have $k = n$, since in this case $\mathsf{threats}(\pi_n) \neq \varnothing$. Thus, this Λ is a

threat in π_k for some $k < n$. Since $\mathsf{threats}(\pi_n) = \varnothing$, let k' with $k < k' \leq n$ be minimal with the property $\Lambda \notin \mathsf{threats}(\pi_{k'})$. By Defintions 2.5.3 and 2.5.5, this can only mean that $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{A}_{k'}]$ is a threat resolution move. Let us redefine $\mathcal{C} = \mathcal{A}_{k'}$, so this argumentation line becomes $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}]$. Again by Def.2.5.5, this argumentation line is in $\mathcal{T}^*_{\mathbb{P}\oplus\pi_{k'}}(\mathcal{A})$. Moreover, by the first claim shown in this Lemma, $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}] \in \mathcal{T}_{\mathbb{P}\oplus\pi_{k'}}(\mathcal{A})$. And using $(n - k')$ applications of Lemma 2.7.2, we conclude that $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}] \in \mathcal{T}_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$. Thus for the present condition (ii), it only remains to prove that this $\mathcal{C}$ is undefeated in $\mathcal{T}_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$.

Assume the contrary, towards a contradiction, i.e that $\mathcal{C}$ is defeated in $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}] \in \mathcal{T}_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$. We show that this gives rise to threat resolution moves of arbitrary length extending $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}]$. Define $\Lambda_1 = \Lambda = [\mathcal{A}, \ldots, \mathcal{B}]$ and $\Lambda_2 = [\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}]$. Since $\mathcal{C}$ is defeated, some defeater $\Lambda_3 = [\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}, \mathcal{D}]$ exists in $\mathcal{T}^*_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$ (with such $\mathcal{D}$ undefeated). Say that Λ_3 occurs in $\mathcal{T}_{\mathbb{P}\oplus\pi_{k''}}(\mathcal{A})$ for some k'' with $k' \leq k'' \leq n$ minimal with this property. By Def. 2.5.3 or Def. 2.5.5 (depending on the type of $\mathcal{A}_{k''}$), we will have that $\Lambda_3 \in \mathcal{T}_{\mathbb{P}\oplus\pi_{k''}}(\mathcal{A})$. Thus, Λ_3 is in $\mathsf{threats}(\pi_{k''})$. The assumption $\mathsf{threats}(\pi_n) = \varnothing$ again implies that some k''' with $k'' < k''' \leq n$ exists which is minimal with the property $\Lambda_3 \notin \mathsf{threats}(\pi_{k'''})$. Thus, if we define $\mathcal{E} = \mathcal{A}_{k'''}$, then this $\mathcal{A}_{k'''}$ denotes a threat resolution move $\Lambda_4 = [\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}]$. We cannot have that this $\mathcal{E}$ is undefeated in $\Lambda_4 \in \mathcal{T}_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$, since this would contradict that $\mathcal{D}$ is undefeated. In consequence some defeater $\mathcal{F}$ exists for it. From this point onwards, the previous reasoning can be arbitrarily repeated. But this is impossible, since π_n has at most n threat resolution moves. From this we conclude that $\mathcal{C}$ is undefeated in $[\mathcal{A}, \ldots, \mathcal{B}, \mathcal{C}] \in \mathcal{T}_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$.

(iii) Let $[\mathcal{A}, \ldots, \mathcal{C}]$ be an arbitrary argumentation line of odd length in $\mathcal{T}^*_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$, and let $[\mathcal{A}, \ldots, \mathcal{C}, \mathcal{D}]$ exist in $\mathcal{T}_{\mathbb{P}\oplus\pi_n}(\mathcal{A})$. Moreover, let $k < n$ be minimal with the property $[\mathcal{A}, \ldots, \mathcal{C}] \in \mathcal{T}^*_{\mathbb{P}\oplus\pi_k}(\mathcal{A})$. In addition, some k' with $k \leq k' < n$ will also exist which is minimal with the property $[\mathcal{A}, \ldots, \mathcal{C}], [\mathcal{A}, \ldots, \mathcal{C}, \mathcal{D}] \in \mathcal{T}^*_{\mathbb{P}\oplus\pi_{k'}}(\mathcal{A})$. An easy induction proof using Definitions 2.5.3 and 2.5.5 shows that $[\mathcal{A}, \ldots, \mathcal{C}]$ and $[\mathcal{A}, \ldots, \mathcal{C}, \mathcal{D}]$ are also in $\mathcal{T}^*_{\mathbb{P}\oplus\pi_n}$. □

Let us remark that using these proofs, we can justify the $\sqsubseteq$-relations from Figure 2.4 (repeated below). Indeed, Lemma 2.7.2 shows the horizontal $\sqsubseteq$-relations in the bottom line. The first claim of Lemma 2.7.4 shows the vertical $\sqsubseteq$-relations. Finally, each $\sqsubseteq$-relation in the top line is easily seen by an inductive proof using Definitions 2.5.3 and 2.5.5.

$$\begin{array}{ccccccc}
\mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}) & \sqsubseteq & \ldots & \sqsubseteq & \mathcal{T}^*_{\mathbb{P}\oplus\pi'}(\mathcal{A}) \\
\sqcap\!| & & \sqcap\!| & & \sqcap\!| \\
\mathcal{T}_{\mathbb{P}\oplus\pi}(\mathcal{A}) & \sqsubseteq & \ldots & \sqsubseteq & \mathcal{T}_{\mathbb{P}\oplus\pi'}(\mathcal{A})
\end{array}$$

Theorem 2.7.5 (Soundness of t-DeLP plan search.)**.** *Let π be the output of the BFS algorithm in the space of plans for $\mathtt{M}$. Then π is a solution for $\mathtt{M}$.*

Proof. Let $\pi = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ be the output of the plan search algorithm in the space of plans for $\mathtt{M}$. Thus, we have that $\mathsf{threats}(\pi) = \varnothing$ and $\mathsf{goals}(\pi) = \varnothing$. Using the π_k-notation, π_n will denote π; and π_0 will denote $\pi_\varnothing$. Then, to prove the claim that $G \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A(\pi))$ it suffices to show the (stronger) claim

$$\bigcup_{0 \leq k \leq n} \mathsf{goals}(\pi_k) \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A(\pi))$$

$$\text{since} \qquad G = \mathsf{goals}(\pi_0) \subseteq \bigcup_{0 \leq k \leq n} \mathsf{goals}(\pi_k)$$

Moreover, by Lemma 2.3.5, we only need to check that, for any literal $\langle \ell, t \rangle$ and plan π_k

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

where $A(t) = \{\, \mathsf{e} \in A(\pi) \mid t_\mathsf{e} \leq t \}$ denotes the actions in the plan whose effects will occur before or at t (recall $\mathsf{post}(\mathsf{e}) = \langle \mu_\mathsf{e}, t_\mathsf{e} \rangle$). The reason is that (finitely-many applications of) Lemma 2.3.5 imply the equivalence

$$\langle \ell, t \rangle \in \mathsf{warr}((\Pi, \Delta) \diamond A(t)) \qquad \text{iff} \qquad \langle \ell, t \rangle \in \mathsf{warr}((\Pi, \Delta) \diamond A(\pi))$$

We proceed to prove the above claim $(\star)$ for goals $\langle \ell, t \rangle$ by induction on t; this is proved together with the auxiliary claim $(\star 2)$ that each action in $A(t)$ is executable:

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

(Base Case $t = 0$) Since actions are durative, we have that $A(0) = \varnothing$. Using this identity, claim $(\star 2)$ can be seen as follows:

$$\begin{array}{rlcll} & (\Pi \oplus A(0), \Delta) & = & (\Pi \cup \mathsf{post}[\varnothing], \Delta) & = \\ = & (\Pi \cup \varnothing, \Delta) & = & (\Pi, \Delta) & = \\ = & (\Pi, \Delta) \diamond \varnothing & = & (\Pi, \Delta) \diamond A(0) & \end{array}$$

For $(\star)$, the proof of the present Base Case for the goal is analogous to that for the Ind. Case, just replacing $\langle \ell, 0 \rangle$ (and $A(0)$) by $\langle \ell, t+1 \rangle$ (and $A(t+1)$), and using the latter claim $(\star 2)$ for the Base Case.

(Ind. Case $t \Rightarrow t+1$) Assume (Ind. Hyp.) that for each $t' \leq t$, and each goal of the form $\langle \ell', t' \rangle \in \bigcup_{0 \leq k \leq n} \mathsf{goals}(\pi^k)$ with $t' \leq t$, we have

$(\star)$ $\langle \ell', t' \rangle \in \mathsf{warr}((\Pi, \Delta) \diamond A(t))$,

$(\star 2)$ $(\Pi \oplus A(t'), \Delta) = (\Pi, \Delta) \diamond A(t')$

We prove that $(\star)$ and $(\star 2)$ hold for $t+1$ as well. Let us show claim $(\star 2)$ first. Let E_{t+1} be the set of actions in $A(\pi)$ with an effect of the form $\langle \cdot, t+1 \rangle$, so we have $A(t+1) = A(t) \cup E_{t+1}$.

$$
\begin{array}{lll}
 & (\Pi \oplus A(t+1), \Delta) & \\
= & (\Pi \oplus (A(t) \cup E_{t+1}), \Delta) & \\
= & (\Pi \cup \mathsf{post}[A(t) \cup E_{t+1}], \Delta) & \\
= & (\Pi \cup \mathsf{post}[A(t)] \cup \mathsf{post}[E_{t+1}], \Delta) & \\
= & ((\Pi \oplus A(t)) \oplus E_{t+1}, \Delta) & \\
= & ((\Pi \oplus A(t)), \Delta) \diamond E_{t+1} & \text{since } \mathsf{pre}[E_{t+1}] \subseteq \mathsf{warr}((\Pi, \Delta) \diamond A(t)) \\
 & & \qquad = \mathsf{warr}(\Pi \oplus A(t), \Delta) \\
= & ((\Pi, \Delta) \diamond A(t)) \diamond E_{t+1} & \text{by the (Ind. Hyp.)} \\
= & (\Pi, \Delta) \diamond (A(t) \cup E_{t+1}) & \text{Lemma 2.3.5, since } A(t) \text{ precedes } E_{t+1} \\
= & (\Pi, \Delta) \diamond A(t+1) &
\end{array}
$$

For the $(\star)$ claim, let $\langle \ell, t{+}1 \rangle$ be an arbitrary goal in $\bigcup_{1 \leq k \leq n} \mathsf{goals}(\pi_k)$. Since $\mathsf{goals}(\pi_n) = \varnothing$, let $k \leq n$ be minimal with the property $\langle \ell, t+1 \rangle \in \mathsf{goals}(\pi_k)$. Since $\langle \ell, t+1 \rangle \notin \mathsf{goals}(\pi_n) = \varnothing$, some k' with $k < k' \leq n$ exists with the property $\langle \ell, t+1 \rangle \notin \mathsf{goals}(\pi_{k'})$. Again, let k' be minimal with this property, so $\langle \ell, t{+}1 \rangle \in \mathsf{goals}(\pi_k) \cap \mathsf{goals}(\pi_{k+1}) \cap \ldots \cap \mathsf{goals}(\pi_{k'-1})$. Consider the cases:

(Case $\mathcal{A}_{k'}$ is a threat resolution move $[\ldots, \mathcal{A}_{k'}]$.) We show that this case is impossible: by definition, threat resolution moves are not regarded as goal-resolving arguments (even in the case where the threat resolving argument has a goal as its conclusion). To see this, note first that Definition 2.5.5 gives:

$$\mathsf{goals}(\pi_{k'}) = \mathsf{goals}(\pi_{k'-1}(\mathcal{A}_{k'})) = (\mathsf{goals}(\pi_{k'-1}) \cup \mathsf{pre}[A^*]) \setminus (\Pi_{\mathsf{f}} \cup \mathsf{OldGoals}(\pi_{k'-1}))$$

By the minimality of k', we have both $\langle \ell, t+1 \rangle \in \mathsf{goals}(\pi_{k'-1})$ and $\langle \ell, t+1 \rangle \notin \mathsf{goals}(\pi_{k'})$, so the above Definition 2.5.5 for $\mathsf{goals}(\pi_{k'})$ implies that $\langle \ell, t+1 \rangle \in \Pi_{\mathsf{f}} \cup \mathsf{OldGoals}(\pi_{k'-1})$. Let us separately consider each set in this union. First, $\langle \ell, t+1 \rangle$ cannot be an old goal in $\pi_{k'-1}$, by the minimality of k w.r.t. $\langle \ell, t+1 \rangle \in \mathsf{goals}(\pi_k)$ and the minimality of $k' > k$ w.r.t. $\langle \ell, t+1 \rangle \notin \mathsf{goals}(\pi_{k'-1})$. On the other hand $\langle \ell, t+1 \rangle$ cannot be in Π_{f} either. The reason is that in this case, Def.2.5.5 for $\mathsf{goals}(\pi_k)$ would give $\langle \ell, t{+}1 \rangle \notin \mathsf{goals}(\pi_k)$, contradicting the assumption that $\langle \ell, t{+}1 \rangle \in \mathsf{goals}(\pi_k)$.

(Case $\mathcal{A}_{k'}$ is an argument step.) From the minimality of k' w.r.t. $\langle \ell, t+1 \rangle \notin \mathsf{goals}(\pi_{k'})$, we have both this property and $\langle \ell, t{+}1 \rangle \in \mathsf{goals}(\pi_{k'-1})$. Combining these facts with the identities (from Definition 2.5.3)

$$
\begin{array}{ll}
\mathsf{goals}(\pi_{k'}) & = \mathsf{goals}(\pi_{k'-1}(\mathcal{A}_{k'})) \\
 & = (\mathsf{goals}(\pi_{k'-1}) \cup \mathsf{pre}[A^*]) \setminus (\{\mathsf{concl}(\mathcal{A}_{k'})\} \cup \Pi_{\mathsf{f}} \cup \mathsf{OldGoals}(\pi_{k'-1}))
\end{array}
$$

we conclude that either $\langle \ell, t{+}1 \rangle = \mathsf{concl}(\mathcal{A}_{k'})$ or $\langle \ell, t{+}1 \rangle \in \Pi_{\mathsf{f}}$ or finally $\langle \ell, t{+}1 \rangle \in \mathsf{OldGoals}(\pi_{k'-1})$. A reason against the latter possibility is, as in the previous case, given by the minimality of oth k and k'. The same can be said against the second possibility $\langle \ell, t{+}1 \rangle \in \Pi_{\mathsf{f}}$, since this would again contradict the assumption

$\langle \ell, t+1 \rangle \in \mathsf{goals}(\pi_k)$. So it remains to check claim ($\star$) for the former possibility. That is, it remains to prove that

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

For this, first we use the assumption $\mathsf{threats}(\pi_n) = \varnothing$ and Lemma 2.7.4, to obtain that

(a) $\mathcal{T}^*_{\mathbb{P} \oplus \pi}(\mathcal{A}_{k'})$ satisfies the conditions for Lemma 2.7.3

Second, the same assumption $\mathsf{threats}(\pi_n) = \varnothing$ implies that for each threat in π_n there is some threat resolution move in π_n. The latter implies, by Def. 2.5.5, that each interfering argument in $\mathcal{T}_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$ has a defeater. This and the previous claim (a) implies that we can apply the second claim of Lemma 2.7.3, to conclude that

(b) $\mathcal{A}'_k$ is undefeated in $\mathcal{T}^*_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$

Finally, (a) implies the main claim in Lemma 2.7.3 (in particular for the argumentation line $[\mathcal{A}_{k'}]$) from which we obtain the equivalence

(c) $\mathcal{A}_{k'}$ is undefeated in $\mathcal{T}^*_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$ iff $\mathcal{A}_{k'}$is undefeated in $\mathcal{T}_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$

Now we reason as follows

$\mathcal{A}_{k'}$ is undefeated in $\mathcal{T}_{\mathbb{P} \oplus \pi_n}(\mathcal{A}_{k'})$	(combining (b) and (c))
$\langle \ell, t+1 \rangle \in \mathsf{warr}(\mathbb{P} \oplus \pi_n)$	(def. of warrant)
$\langle \ell, t+1 \rangle \in \mathsf{warr}(\mathbb{P} \oplus A(t+1))$	(by Lemma 2.3.5)
$\langle \ell, t+1 \rangle \in \mathsf{warr}(\mathbb{P} \diamond A(t+1))$	(be the above claim ($\star 2$) for $t+1$)

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

$\langle \ell, t \rangle \in G$	$\Rightarrow$	$\langle \ell, t \rangle \in \mathsf{warr}(\mathbb{P} \diamond A(t))$	(the above ind. proof)
$\langle \ell, t \rangle \in \mathsf{warr}(\mathbb{P} \diamond A(t))$	$\Rightarrow$	$\langle \ell, t \rangle \in \mathsf{warr}(\mathbb{P} \diamond \pi_n)$	(by Lemma 2.3.5)
$\langle \ell, t \rangle \in G$	$\Rightarrow$	$\langle \ell, t \rangle \in \mathsf{warr}(\mathbb{P} \diamond \pi_n)$	(by transitivity)
G	$\subseteq$	$\mathsf{warr}(\mathbb{P} \diamond \pi_n)$	(since $\langle \ell, t \rangle$ is arbitrary)

□

2.8 Completeness of BFS search for backward t-DeLP planning

Theorem 2.8.1 (Completeness of t-DeLP BFS plan search)**.** *Let $\mathbb{M} = (\mathbb{P}, A, G)$ be a planning domain and assume some solution $A' \subseteq A$ exists. Then, the BFS search in the space of plans terminates with an output π.*

Proof. Let $A' \subseteq A$ be a solution for a given planning domain $\mathtt{M} = (\mathbb{P}, A, G)$, so $G \subseteq \mathsf{warr}(\mathbb{P} \diamond A')$. Without loss of generality, we assume that this set A' is a $\subseteq$-minimal solution: for any proper subset $A'' \subsetneq A'$, we have $G \not\subseteq \mathsf{warr}(\mathbb{P} \diamond A'')$. We proceed to define by induction a plan of the form $\pi_n = \pi_{\varnothing}(\Lambda_1, \ldots, \Lambda_n)$. This will be proved to be a plan in the search space that satisfies the Terminating Conditions and whose set of actions is $A(\pi_n) = A'$. This is enough, since if the planning algorithm terminates with some other output $\neq \pi_n$ before π_n is generated, we are done.

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

Now, a consequence of the $\subseteq$-minimality of A' is that $\mathbb{P} \diamond A' = \mathbb{P} \oplus A'$. This can be seen by the following induction on t. Define $A(t) = \{\mathsf{e} \in A' \mid t_{\mathsf{e}} \leq t\}$. (Base Case $t = 0$.) Since actions in A are durative, we have $A(0) = \varnothing$, which implies $\mathbb{P} \diamond A(0) = \mathbb{P} = \mathbb{P} \oplus A(0)$. (Ind. Case $t \Rightarrow t+1$.) Assume that $\mathbb{P} \diamond A(t) = \mathbb{P} \oplus A(t)$, so $\mathsf{warr}(\mathbb{P} \diamond A(t)) = \mathsf{warr}(\mathbb{P} \oplus A(t))$. In case $A(t+1) \smallsetminus A(t) = \varnothing$, we are done since then $\mathbb{P} \oplus A(t+1) = \mathbb{P} \diamond A(t+1)$ reduces to the Ind. Hyp.; hence, we can assume that $A(t+1) \smallsetminus A(t) = \{\mathsf{f}_1, \ldots, \mathsf{f}_r\}$. Define $E_{t+1} = A(t+1) \smallsetminus A(t)$, and select an arbitrary element $\mathsf{e} \in E_{t+1}$. This implies that its set of preconditions is of the form $\mathsf{pre}(\mathsf{e}) = \{\langle \ell, t \rangle, \ldots, \langle \ell', t \rangle\}$. So by the Ind. Hyp.,

$$\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\mathbb{P} \diamond A(t)) \;\Leftrightarrow\; \mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\mathbb{P} \oplus A(t))$$

On the other hand, the $\subseteq$-minimality of A' implies that $\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\mathbb{P} \diamond A(t))$ –the reason is that otherwise $A' \cup \{\mathsf{e}\}$ would be a solution. The last two facts imply that $\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{warr}(\mathbb{P} \oplus A(t))$. From all this, we conclude

$$(*) \quad \langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle \in (\mathbb{P} \diamond A(t)) \diamond \{\mathsf{e}\} \iff \langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle \in (\mathbb{P} \oplus A(t)) \diamond \{\mathsf{e}\}$$
$$(*2) \quad \langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle \in (\mathbb{P} \oplus A(t)) \diamond \{\mathsf{e}\} \iff \langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle \in (\mathbb{P} \oplus A(t)) \oplus \{\mathsf{e}\}$$

Since e was an arbitrary element of E_{t+1}, we can reason as follows (note that we redefine $\ldots \oplus \{\mathsf{f}\}$ as $\ldots \oplus \mathsf{f}$)

$$\begin{array}{lll}
 & \mathbb{P} \diamond A(t+1) & \\
= & (\mathbb{P} \diamond A(t)) \diamond E_{t+1} & \text{since } A(t) \text{ precedes } E_{t+1} \\
= & (\mathbb{P} \oplus A(t)) \diamond E_{t+1} & \text{(by the Ind. Hyp.)} \\
= & (((\mathbb{P} \oplus A(t)) \oplus \mathsf{f}_1) \diamond \mathsf{f}_2) \cdots \diamond \mathsf{f}_r & \text{(by facts } (*), (*2) \text{ with } \mathsf{e} = \mathsf{f}_1) \\
= & (((\mathbb{P} \oplus A(t)) \oplus \mathsf{f}_1) \oplus \mathsf{f}_2) \cdots \diamond \mathsf{f}_r & \text{(by fact } (*2) \text{ with } \mathsf{e} = \mathsf{f}_2; \\
 & & \qquad \text{and Lemma 2.3.5)} \\
 & \vdots & \vdots \\
= & (((\mathbb{P} \oplus A(t)) \oplus \mathsf{f}_1) \oplus \mathsf{f}_2) \cdots \oplus \mathsf{f}_r & \text{(by fact } (*2) \text{ with } \mathsf{e} = \mathsf{f}_r; \\
 & & \qquad \text{and Lemma 2.3.5)} \\
= & (\mathbb{P} \oplus A(t)) \oplus E_{t+1} & \\
= & \mathbb{P} \oplus A(t+1) &
\end{array}$$

Since A' is a finite union of $A(t)$ sets, this concludes the proof of the claim

$$(\star) \quad \mathbb{P} \oplus A' = \mathbb{P} \diamond A'.$$

We proceed to define the set of plan steps in the desired plan. By simultaneous induction, define first a pair of sets Steps and $\mathsf{Threats}$ as the $\subseteq$-minimal sets containing:

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

Note first that this definition will give a finite set $|\mathsf{Steps}|$, since $\mathsf{Steps} \subseteq \bigcup_{\langle \ell,t \rangle \in G^+} \mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle}$ is included in the union of finitely-many sets (exactly, $|G^+|$), each of which is a finite set $\mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle})$. Now, that this construction can be done is obvious for the former two conditions. For the third, we must show that at least an undefeated defeater $\mathcal{C}$ exists for each interfering argument $[\mathcal{A}_{\langle \ell,t \rangle}, \ldots, \mathcal{B}]$. This is shown by induction, together with the auxiliary claim that each defending argument is undefeated.

(Base Case $[\mathcal{A}_{\langle \ell,t \rangle}, \mathcal{B}] \in \mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle})$.) On the one hand, $\mathcal{A}_{\langle \ell,t \rangle}$ is undefeated by assumption. This and the case assumption imply that this $\mathcal{B}$ is defeated in $[\mathcal{A}_{\langle \ell,t \rangle}, \mathcal{B}] \in$, so an undefeated defeater $\mathcal{C}$ must exist in some argumentation line $[\mathcal{A}_{\langle \ell,t \rangle}, \mathcal{B}, \mathcal{C}]$.

(Ind. Case $[\mathcal{A}_{\langle \ell,t \rangle}, \ldots, \mathcal{A}, \mathcal{B}] \in \mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle})$.) Assume (Ind. Hyp.) that this $\mathcal{A}$ is undefeated in $\mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle})$. In consequence, $\mathcal{B}$ must be defeated in $[\mathcal{A}_{\langle \ell,t \rangle}, \ldots, \mathcal{A}, \mathcal{B}] \in \mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle})$, and so some $[\mathcal{A}_{\langle \ell,t \rangle}, \ldots, \mathcal{A}, \mathcal{B}, \mathcal{C}]$ must exist in $\mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle})$ where $\mathcal{C}$ is undefeated.

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

Finally, we proceed with the inductive definition of a plan of the form

$$\pi_n = \pi_{\varnothing}(\Lambda_1, \ldots, \Lambda_n)$$

$$\text{satisfying} \qquad \begin{array}{cc} \{\Lambda_1, \ldots, \Lambda_n\} = \mathsf{Steps} & \mathsf{goals}(\pi_n) = \varnothing \\ A(\pi_n) = A' & \mathsf{threats}(\pi_n) = \varnothing \end{array}$$

(Base Case π_0).
Clearly, $\pi_0 = \pi_\varnothing$ is a plan for $\mathtt{M}$.

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

(Sub-case $\mathsf{threats}(\pi_m) \neq \varnothing$) Let then $\Lambda = \Lambda^{2k+2} = [\mathcal{A}_1, \ldots, \mathcal{A}_{2k+1}, \mathcal{A}_{2k+2}]$ be a threat in π_m, where $\mathcal{A}_1$ is some argument step $\mathcal{A}_1 = \mathcal{A}_{\langle \ell, t\rangle} \in \mathsf{ArgSteps}$. Let us denote the initial fragments of this threat Λ as $\Lambda^i = [\mathcal{A}_1, \ldots, \mathcal{A}_i]$.

Now, by definition of the sub-tree $\mathcal{T}^*(\mathcal{A}_1)$, some defending argument $\mathcal{C}$ exists in some $\Lambda^{2k+3} = \Lambda^\frown[\mathcal{C}] \in \mathsf{Steps}$. Let $A^* \subseteq A'$ be the (unique) $\subseteq$-minimal set with the property $\mathsf{base}(\mathcal{C}) \subseteq \Pi \cup \mathsf{post}[A(\pi_m) \cup A^*]$; and also let $\mathcal{C}^- = \mathcal{C} \setminus \mathsf{base}(\mathcal{C})$. We check conditions (i)-(v) from Def. 2.5.5 for those $\mathcal{C}^-$ and A^*:

(i) $A(\pi_m) \cup A^*$ *is non-overlapping*; the claim holds because this set is a subset of A', which by assumption is non-overlapping.

(ii) $(\Pi \oplus (A(\pi_m) \cup A^*), \Delta)$ *is a t-DeLP program*; this is a direct consequence from the fact that $(\Pi \oplus A', \Delta)$ is a t-DeLP program and $A(\pi_m) \cup A^* \subseteq A'$.

(iii) $\mathcal{C}^- \cup \mathsf{base}(\mathcal{C}^-) (= \mathcal{C})$ *is an argument in this program*; this follows from: (1) $\mathcal{C}^- \subseteq \Pi_\mathsf{r} \cup \Delta$, (2) $\mathsf{base}(\mathcal{C}^-) \subseteq \Pi_\mathsf{f} \oplus (A(\pi_m) \cup A^*)$ and (3) that $\mathcal{C}$ is an argument in $\mathbb{P} \oplus A'$; these facts imply that the conditions (1)-(4) from Def. 1.3.5 are preserved from $\mathbb{P} \oplus A'$ to $(\mathbb{P} \oplus A(\pi_m)) \oplus A^*$. Also the second claim that $\Lambda^\frown[\mathcal{C}]$ *is an argumentation line for* $\mathcal{A}_1$ *in* $\mathbb{P} \oplus (A(\pi_m) \cup A^*)$ holds. The facts (1) Λ is an arg. line in this program, (2) $\mathcal{C}$ is an argument in this program, and (3) $\Lambda^\frown[\mathcal{C}]$ is an argumentation line in the extended program $\mathbb{P} \oplus A'$ imply the preservation of the claim from $\mathbb{P} \oplus A'$ to $(\mathbb{P} \oplus A(\pi_m)) \oplus A^*$.

(iv) A^* *is* $\subseteq$*-minimal with* (iii); this follows from the above definition of A^*.

(Sub-case $\mathsf{threats}(\pi_m) = \varnothing$ and $\mathsf{goals}(\pi_m) \neq \varnothing$) Let $\langle \ell, t\rangle \in \mathsf{goals}(\pi_m)$. In case $\langle \ell, t, \rangle \in G$, some argument step $[\mathcal{A}_{\langle \ell, t\rangle}] \in \mathsf{ArgSteps} \subseteq \mathsf{Steps}$ exists for $\langle \ell, t\rangle$. On the other hand, if $\langle \ell, t\rangle \notin G$, then by the definition of $\mathsf{goals}(\cdot)$ in Defs. 2.5.3 and 2.5.5, we must have $\langle \ell, t\rangle \in \mathsf{pre}(\mathsf{e})$ for some $\mathsf{e} \in A(\pi_m) \subseteq A'$, in which case by definition we have $[\mathcal{A}_{\langle \ell, t\rangle}] \in \mathsf{Steps}$. Let us denote by $\mathcal{A}$ this argument $\mathcal{A}_{\langle \ell, t\rangle}$, and also let A^* be the unique $\subseteq$-minimal set of actions satisfying $\mathsf{base}(\mathcal{A}) \subseteq \Pi \cup \mathsf{post}[A(\pi_m) \cup A^*]$. Conditions (i), (ii) and (iv) from Def. 2.5.3 are proved as in the previous case. We check the remaining conditions (iii) and (v).

(iii) $\mathcal{A}$ *is an argument in* $(\mathbb{P} \oplus A(\pi_m) \oplus A^*$; this follows from: $\mathcal{A} \subseteq (\Pi \oplus (A(\pi_m) \cup A^*)) \cup \Delta$ and that $\mathcal{A}$ is an argument in $(\Pi \oplus A', \Delta)$; as in the previous case the latter fact preserves conditions (1)-(4) from Def. 1.3.5.

(v) *that* $\mathsf{pre}[A^*] \cup \mathsf{literals}(\mathcal{A})$ *is consistent with previous plans' goals and the literals of argument steps*; for this, note that by construction, $\mathsf{ArgSteps}$ is

a set of argument steps $\mathcal{A}_{\langle \ell,t \rangle}$ undefeated in $\mathcal{T}_{\mathbb{P} \diamond A'}(\mathcal{A}_{\langle \ell,t \rangle})$. Thus, by the Sub-Arguments postulate, the sub-arguments for arbitrary literals in these arguments are also undefeated. Hence, all these literals are warranted in $\mathbb{P} \oplus A'$. The literals from $\mathsf{goals}(\pi_k)$ with $k \leq n$ or $\mathsf{pre}[A^*]$ are also warranted in $\mathbb{P} \oplus A'$, since they are in $G \cup \mathsf{pre}[A']$. Hence, by the Direct Consistency postulate all these are consistent.

(Sub-case $\mathsf{threats}(\pi_m) = \varnothing$ and $\mathsf{goals}(\pi_m) = \varnothing$) In this case, we want to show that $A(\pi_m) = A'$. We have

- π_m is a plan for $\mathbb{M}$ (by the Ind. Hyp.)
- π_m satisfies the Terminating Condition (by the Sub-Case assumption),

Thus, we can apply the soundness theorem 2.7.5, and conclude that π_m is a solution for $\mathbb{M}$. Clearly, by construction of $A(\pi_m)$ from Steps (i.e. from A'), we have that $A(\pi_m) \subseteq A'$. This, together with the facts that $A(\pi_m)$ is a solution and that A' is a $\subseteq$-minimal solution implies that $A(\pi_m) = A'$.

□

2.9 Conclusions and Related Work

In the present chapter, we adapted the notion of state transition systems to the case where t-DeLP is used as the underlying logic. A state, in a t-DeLP state transition system, is a partial description of a world-line. This state is identified as the t-DeLP logical closure of a mutex logical program (as defined in Chapter 1).

The planning problems that can be expressed in this framework consist of a logical program (representing the planner's beliefs), temporal actions which the executing agents might try, and temporal goals. Two planning systems were considered: a rather simple planning system for forward planning; and a more complex planning system for a backward approach to the generation of plans. The main contribution is the study of Breadth First Search as an algorithm for centralized planning. This is shown to be sound and complete in each of the two t-DeLP planning systems proposed, for forward and backward search.

The advantages of a t-DeLP based planner, in comparison to temporal logics about actions or temporal planners in the literature, are mainly those inherited from t-DeLP logic programming (Chapter 1). These advantages consist in natural representations for common-sense reasoning; and a powerful inference system for temporal reasoning of this kind.

The combination of t-DeLP logic programming with planning techniques is largely inspired by the DeLP-based partial order planning (POP) system in [62]. While this planning system is more flexible, due to the use of partially ordered plans, the underlying logic is less expressive, given the implicit time approach and hence the absence of temporal reasoning. In addition, we suggest some natural simplifications on the representation of actions used in [62], which are

partly inspired by those in [111]. The resulting actions greatly simplify the different types of plan threats that can occur and, correspondingly, the types of threat resolution moves to be defined against them.

In a broader context, the literature on temporal planning is quite rich, though most proposals are based on a monotonic (and often simple) logical inferences. As a consequence, the resulting planning systems are unable to address the ramification problem in full generality. In this category, a number of temporal planning system have been proposed (see [66], Ch. 14). Among them, some planners combine can plan space planning (for temporal actions) with CSP techniques for constraint satisfaction of the temporal constraints. See also Chapter B in the Appendix.

In the literature on logic, [96] shows that planning with linear temporal logic LTL can be reduced to SAT problems. Some other powerful temporal logics exist, like CTL* [56], which do not address practical reasoning on their own.

The present chapter is also related to logics of actions and logics of time, provided the t-DeLP planning system includes a state transition system which can be used for the purpose of reasoning about actions. In this case, similar remarks as those above for temporal planners can be made for these logics. For the particular case of the Propositional Dynamic Logic PDL, see for example [152] for a solution of the frame problem in the expressive PDL formalism.

In contrast, there is not much literature on planning systems whose underlying logic is non-monotonic. To our knowledge, the present contribution is the first proposal combining temporal defeasible logic with planning.

Chapter 3

Multi-planner Dialogues for cooperative planning in t-DeLP

3.1 Introduction

In this chapter we study a decentralized version of the backward planning algorithm from Chapter 2. Informally, the new algorithm takes the form of a dialogue among executing agents who wish to find a joint plan for a fixed set of goals. Moreover, this joint plan should be agreed upon by all the agents. These dialogues will consist of a set of rounds, with agents taking turns at each round; at the corresponding turn, the agent will send data to the agent next in line. See Figure 3.1 for an illustration of these dialogues. Without a central planner (as it was assumed in Chapter 2), these executing agents must also take the the role of planners (as well as communicative agents). Being planners themselves, the agents can contribute to the dialogue by generating their own proposals for plans and evaluating those from other agents. In the present chapter, we will study a dialogue protocol for this kind of distributed planning problems.

The present focus, in particular, is on cooperative scenarios where these planner-executing agents share the same set of goals. In contrast, the agents' abilities (actions) and knowledge (of the initial state or rules) can differ significantly among them, at least at the start of the discussion. In the proposed framework, then, each planner-executing agent is initially endowed with a planning domain, so a distributed planning problem will be defined by a collection of these planning domains.

The formal study of the proposed dialogue-based planning algorithm will be done in comparison with the centralized method from Chapter 2. To this end, a distributed planning problem (a tuple of planning domains) is translated to a centralized planning problem (a planning domain) simply by gathering all the

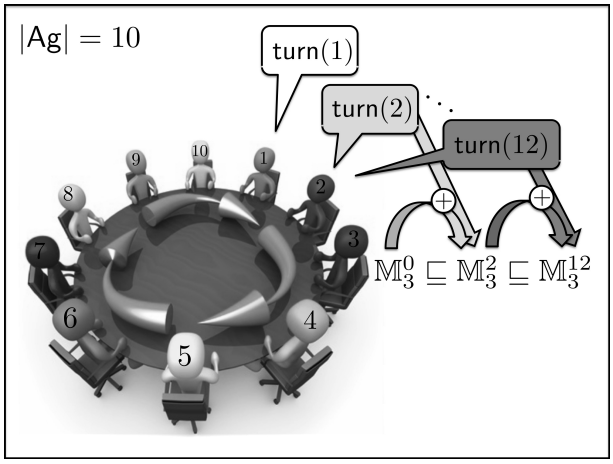


Figure 3.1: A representation of the cyclic dialogues for cooperative planning among a set of agents $\mathsf{Ag} = \{1, \ldots, 10\}$, starting with initial planning domains $\langle \mathbb{M}_a^0 \rangle_{a \in \mathsf{Ag}}$ at turn 0. Here, agent 1 communicates to agent 2, 2 to 3, etc. and agent 10 to agent 1. Note, for example, that agent 2 speaks at turns $2, 12, 22, \ldots$, each time causing an expansion in the planning domains of agent 3.

information of the agents into a unique planning domain for a central planner. Ideally, solutions agreed upon in a distributed algorithm should be as good as those obtained from a centralized planning domain.

Briefly, the motivations for a decentralized approach multi-agent t-DeLP planning is *to enable a set of agents* Ag *with the same interests to*

propose, discuss and reach agreements upon joint plans

- (1) without a central planner
- (2) by sharing relevant information only
- (3) within the planning phase, (rather than by merging agents' individual plans)

Thus, the main motivation for a decentralized approach to t-DeLP (backward) planning, lies beyond (or besides) the usual reasons for decentralized algorithms, namely, computational efficiency. Agents are now assumed to be autonomous entities: they reason and act for internal reasons (or towards their own goals); and in case these abilities prove insufficient or too costly, these agents can make use of some social abilities to better promote these goals.

Second, centralized methods for autonomous agents might initially involve massive communication costs, depending on the size of agents' planning domains; moreover, most of the communicated data might simply be not relevant for the planning problem at hand.

Finally, we consider the discussion about plans to occur during the planning phase. This is not the only approach in the literature, post-planning dialogues

for the merging of different agents' plans have also been studied. The reason for this is mainly technical: the logical interdependencies between actions and arguments suggest to discuss a plan after each refinement during its construction of a plan.

Before offering an informal introduction to the technical aspects of the proposed dialogues, let us expand on the cooperative reading of distributed planning problems. The practical autonomy of agents assumed above can be considered with or without a similar autonomy at the level of goals; that is, whether an agent decides which goals to pursue. The proposed framework is informally presented under this stronger notion of autonomy, so distributed planning is interpreted as cooperative planning. Alternatively, the agents might not be autonomous w.r.t. goals, but simply promote the goals of some external user. Under this alternative reading, distributed planning can be understood as problems in collaboration.

In a sketch, the proposed dialogue-based algorithm for plan search is as follows. Each agent is initially endowed with a planning domain $\mathbb{M}_a = ((\Pi_a, \Delta_a), A_a, G)$, containing the believed facts and rules (Π_a, Δ_a); a list A_a of actions available to this agent a; and a set of common goals G. The dialogue is essentially an exchange of proposals for plan refinements or threats upon a plan. These proposals are possibly incomplete plan steps or threats, to be completed or challenged by other agents.

Rather than considering a (turn-based) public discussion, at each turn, the corresponding agent sends a message only to the next agent in line. This does not prevent that, during the dialogue, the agents keep continuously expanding their planning domains with new facts, rules and actions contributed by any other agent, e.g. in the latter's own plan proposals. In addition to threats, irrevocable arguments against a plan step must be communicated separately. Many potential pre-arguments in these proposals, thus, will not lead to any real or interesting plan step or threat. But this exchange of (only) potentially relevant information is the price to pay for the completeness of search algorithms under the present distributed approach to t-DeLP planning.

Finally, let us remark that only a decentralized approach for backward t-DeLP planning will be considered. Again, a forward approach is rather trivial in comparison, in the sense that a dialogue for forward planning (based on Algorithm 1) would essentially decompose into a set of dialogues for query-answering in t-DeLP, one for each generated plan. (The latter problems were studied in [135]; see also the related work in Section 3.5.)

Structure of the Chapter

First, a formal description of the problem of distributed plan search is introduced in Section 3.2. Then, in Section 3.3, we present the type of dialogue proposed to solve the present problem and show it is well-defined. Finally, in Section 3.4 we provide soundness and completeness theorems for a planning algorithm based on such dialogues. Section 3.5 contains a summary of conclusions and a description of the Related Work.

3.2 Distributed and centralized planning domains

In this section we introduce multiple-planner versions of the definitions found in the previous chapter. For example, each agent $a \in \mathsf{Ag}$ is now endowed with an initial planning domain $\mathbb{M}_a$. As we said, we will compare the present distributed methods with centralized planning, the latter implemented using the single-planner techniques from Chapter 2.

Definition 3.2.1 (Multi-planner domain; Union of planning domains; Centralized planning)**.** Given a language TLit and a set of planner agents $\mathsf{Ag} = \{1, \ldots, m\}$, let $\mathbb{M}_a = ((\Pi_a, \Delta_a), A_a, G)$ be a planning domain in TLit for each agent $a \in \mathsf{Ag}$. Then, we say $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$ is a *multi-planner domain*, if $\bigcup_a \Pi_a$ is a consistent set of literals. We also define the component-wise *union* of two planning domains, say $\mathbb{M}_1, \mathbb{M}_2$, as follows

$$\mathbb{M}_1 \sqcup \mathbb{M}_2 = ((\Pi_1 \cup \Pi_2, \Delta_1 \cup \Delta_2), A_1 \cup A_2, G)$$

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

$$\mathbb{M}_{\mathsf{Ag}} \;=\; \bigsqcup_{a \in \mathsf{Ag}} \mathbb{M}_a \;=\; \mathbb{M}_1 \sqcup \ldots \sqcup \mathbb{M}_{|\mathsf{Ag}|}$$

The present focus on collaborative planning can be generalized to altruistic cooperation, if other agents' goals are added to each one's list of goals (if jointly consistent). For example, we would define $\mathbb{M}_1 \sqcup \mathbb{M}_2 = ((\cdot, \cdot), \cdot, G_1 \cup G_2)$, thus making Definition 3.2.1 a particular case of it with $G_1 = G_2$. The techniques presented in this chapter suffice for altruistic planning. (For a comparison with the most general case of self-interested multiple-agent planning, see the related work in Section 3.5.)

Definition 3.2.2 (Expansion)**.** Let $\mathbb{M} = ((\Pi, \Delta), A, G)$ and $\mathbb{M}' = ((\Pi', \Delta'), A', G')$ be planning domains. We say $\mathbb{M}'$ is an *expansion* of $\mathbb{M}$, denoted $\mathbb{M} \sqsubseteq \mathbb{M}'$, iff for each component $Y \in \{\Pi, \Delta, A, G\}$ of $\mathbb{M}$, its counterpart Y' extends Y, i.e. $Y \subseteq Y'$.

Notice in particular that for any pair $\mathbb{M}_1, \mathbb{M}_2$ we have that $\mathbb{M}_1, \mathbb{M}_2 \sqsubseteq \mathbb{M}_1 \sqcup \mathbb{M}_2$.

The initial differences among agents' planning domains constitute a new problem with respect to the single-planner case of Chapter 2. The problem is that agents need not agree upon the following questions:

(1) whether a given plan step $\mathcal{A}$ exists,

(2) whether a sequence $\pi = \pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ actually defines a plan, or

(3) which plan does this π define (which threats exist, or open goals remain)

Disagreements of these kinds are caused, respectively, by: (1) agents ignoring some of the elements of $\mathcal{A}$ (its rules, and the initial facts or actions supporting its base); (2) by agents ignoring strict information which either can (and should) replace this plan step, or which contradicts it; and (3) by ignoring some elements for an existing threat to some step $\mathcal{A}_k$ in π; or by ignoring that some open goal is actually a strict fact as well (hence, a solved goal). Even worse, all the agents might simultaneously believe that a newly made proposal π is a plan, and be wrong; and similarly, whether a newly generated plan π has no threats. Only after enough discussion turns on this question, the agents can decide about these questions.

A disagreement about an acknowledged plan $\pi = \pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ stems from the fact that π gives rise to different triples (*actions, trees, goals*) when interpreted by different planning domains (e.g. by different agents, or by an agent at different turns). For this reason, from here on we introduce a superscript notation for interpreted plans $\pi^{\mathbb{M}}$ and distinguish between:

- a plan π, simply denoting a sequence of plan steps

  $$\pi = \pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$$

  i.e., abstracting from any particular $\mathbb{M}$ for which π is actually a plan, and

- an interpreted plan $\pi^{\mathbb{M}}$ (or, simply, a plan), denoting the particular result of computing π in some planning domain $\mathbb{M}$ using Definitions 2.5.1, 2.5.3 and 2.5.5

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

Later we will add to these concepts from Chapter 2 a new kind of interpretations of plans π, namely

- a freely interpreted plan $\pi^{(\mathbb{M})^+}$; this is similar to $\pi^{\mathbb{M}}$ but with dialectical trees also containing potential threats and a claim on which goals remain open; again it is a tuple

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

3.3 Turn-based Dialogues for Cooperative Planning in t-DeLP

The dialogues will consist in a series of rounds among a set of agents $\mathsf{Ag} = \{1, \ldots, r\}$. Each agent speaks once in each round, and always to the same agent. After an initial empty turn 0, the dialogue starts at turn 1 with agent $1 \in \mathsf{Ag}$; the next agent to speak is simply computed as

$$f(m+1) = \begin{cases} f(m) + 1 & \text{if } f(m) < |\mathsf{Ag}| \\ 1 & \text{if } f(m) = |\mathsf{Ag}| \end{cases}$$

At the current turn $m+1$, then, the speaking agent is $f(m+1) \in \mathsf{Ag}$, who communicates a tuple $\mathsf{turn}(m)$ to the next agent $f(m+1)$. The tuple $\mathsf{turn}(m)$ has the following elements:

$$(\,\mathsf{Preplans}_m,\, \mathsf{Plans}_m,\, \mathsf{Trueplans}_m,\, \mathsf{Data}_m\,)$$

For a given planning domain $\mathbb{M}^m = (\mathbb{P}^m, A^m, G)$, these components represent the following sets

- $\mathsf{Preplans}_m$ = the set of incomplete uninterpreted plans, also denoted $\pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A})$ or $\pi(\mathcal{A})$; this $\mathcal{A}$ is an incomplete argument in the program $(\mathbb{P}^m \oplus \pi_n) \oplus A^*$
- Plans_m = a set of pairs $(\pi, \pi^{(\mathbb{M}^m)^+})$, where π is an uninterpreted plan $\pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$; and $\pi^{(\mathbb{M}^m)^+}$ is the free interpretation of π in $\mathbb{M}^m$
- $\mathsf{Trueplans}_m$ = an interpreted plan $\pi^{\mathbb{M}^m}$ in a planning domain $\mathbb{M}^m$, that is, a tuple *(actions, trees, goals)* computed using Definitions 2.5.3 and 2.5.5; this interpretation of π in $\mathbb{M}^m$ is presumed to be the correct one.
- Data_m = a set of auxiliary information: strict facts showing that some "open goals" are actually solved; or the actions supporting plan steps in pre-plans

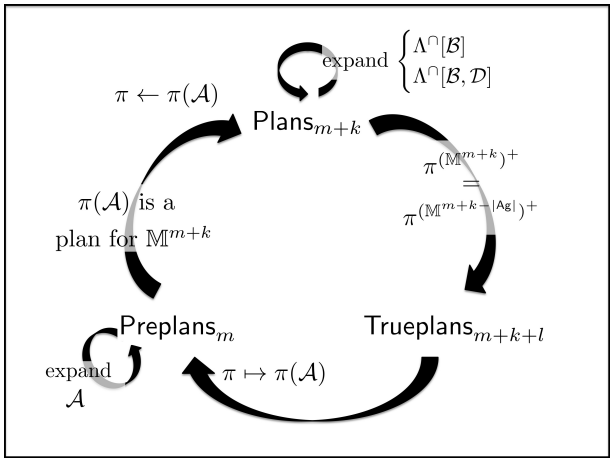


Figure 3.2: A representation of the phases in the turn-based construction of a plan: $\mathsf{Preplans}_m \to \mathsf{Plans}_{m+k} \to \mathsf{Trueplans}_{m+k+l} \to \mathsf{Preplans}_{m+k+l+1}$ and so on. The labels on the corresponding arrows, e.g. $\mathsf{Preplans}_m \to \mathsf{Plans}_{m+k}$, describe the conditions for such a transition to occur, e.g. that a pre-plan ha been completed into a plan. In case this condition fails, following the example, $\pi(\mathcal{A})$ will stay as a pre-plan $\mathsf{Preplans}_{m+k-1} \to \mathsf{Preplans}_{m+k}$, and agent $f(m+k)$ will try to expand $\mathcal{A}$ into a plan step.

The free interpretation of a plan in some planning domain $\mathbb{M}^m$ does include the (usual) interpretation $\pi^{\mathbb{M}^m}$ plus existing pre-threats (according to $\mathbb{M}^m$). The single-planner notion of interpreted plan we considered in Chapter 2 corresponds to the elements sets of the form $\mathsf{Trueplans}_m$.

Definition 3.3.1 (Pre-plan)**.** Given a planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$, and a plan π for $\mathbb{M}$, let $\mathcal{A} \subseteq \Pi \cup \Delta \cup \mathsf{post}[A]$ be arbitrary and define $A^* = \{\mathsf{e} \in A \smallsetminus A(\pi) \mid \langle \mu_{\mathsf{e}}, t_{\mathsf{e}} \rangle \in \mathsf{base}(\mathcal{A})\}$. Then, we say that $\pi(\mathcal{A})$ is a *pre-plan for* $\mathbb{M}$ iff conditions (i)-(v) from the Def. 2.5.3 hold for $\mathcal{A}^- = \mathcal{A} \smallsetminus \mathsf{base}(\mathcal{A})$ and A^*, when (ii)-(iii) are replaced by:

(ii) $((((\Pi \oplus \pi) \oplus A^*) \cup (\mathsf{base}(\mathcal{A}) \smallsetminus \mathsf{post}[A]), \Delta)$ is a t-DeLP program

(iii) $\mathcal{A}$ is an argument in this program

We also say that $\pi([\mathcal{A}_k, \ldots, \mathcal{A}_m, \mathcal{B}, \mathcal{A}])$ is a *pre-plan* for $\mathbb{M}$ iff $[\mathcal{A}_k, \ldots, \mathcal{A}_m, \mathcal{B}]$ is a threat, and conditions (i)-(iv) from Def. 2.5.5 hold if we replace (ii) and (iii) as follows:

(ii) $(((\Pi \oplus \pi) \oplus A^*) \cup (\mathsf{base}(\mathcal{A}) \smallsetminus \mathsf{post}[A]), \Delta)$ is a t-DeLP program

(iii) $\mathcal{A}$ is an argument in this program and $\sim\mathsf{concl}(\mathcal{A}) \in \mathsf{literals}(\mathcal{B})$

The tuples (actions, trees, goals) as defined in Defs. 2.5.3 and 2.5.5 are left undefined for pre-plans.

In this sense, pre-plans are terminal fragments of conceivable plan steps, which might be completed by other agents into plans.

When a pre-plan $\pi(\mathcal{A})$ is completed (at turn m) into an apparent plan, say, $\pi(\mathcal{A}^+)$ for some $\mathcal{A}^+ \supsetneq \mathcal{A}$, the resulting plan $\pi(\mathcal{A}^+)$ is communicated as an element in the set Plans_m. At this stage, agents communicate interpreted plans, that is, tuples (actions, trees, goals). The evaluation of such a plan is done by communicating its pre-threats and solved goals. Pre-threats, similarly to pre-plans, are terminal fragments of possible threats. They are only required to attack the corresponding plan step, rather than be a defeater for it, or constitute with it an argumentation line.

Definition 3.3.2 (Pre-threat)**.** Let a plan $\pi = \pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ for some planning domain $\mathbb{M} = (\mathbb{P}, A, G)$ be given. We say that $\mathcal{B} \subseteq \mathbb{P} \oplus \pi$ is a *pre-threat* for the plan step $[\mathcal{A}_n]$ or $[\mathcal{A}_k, \ldots, \mathcal{A}_n]$ iff

(i) $(\mathbb{P} \oplus \pi) \cup (\mathsf{base}(\mathcal{B}) \smallsetminus \mathsf{post}[A])$ is a t-DeLP program, and

(ii) $\mathcal{B} \cup \mathsf{base}(\mathcal{B})$ is an argument in this program and $\sim\mathsf{concl}(\mathcal{B}) \in \mathsf{literals}(\mathcal{A}_n) \smallsetminus \mathsf{base}(\mathcal{A}_n)$

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

In particular, a pre-threat can eventually become a threat for $\mathcal{A}_k$, in which case it will be included in the usual sub-tree $\mathcal{T}^*_{\mathbb{P}\oplus\pi}(\mathcal{A}_k)$. In addition to the above pre-threats, we specifically consider strict pre-threats. These are incomplete strict arguments $\mathcal{D}$ which, if completed with strict information, prove that some presumed argument is not so. There are two types of strict pre-threats, depending on which argument is challenged.

Definition 3.3.3 (Strict threat; Strict pre-threat)**.** Let $\pi_\varnothing(\ldots,[\ldots,\mathcal{A}_n])$ be a plan for $\mathbb{M}$, and let $\mathcal{A}$ be either the last step $\mathcal{A}=\mathcal{A}_n$ or a threat to it, i.e. $[\mathcal{A}_k,\ldots,\mathcal{A}_n,\mathcal{A}]$ is a threat in $\mathbb{M}$. Let $\mathbb{M}\sqsubseteq\mathbb{M}'=((\Pi',\Delta),A',G)$ be such that either of the following holds:

($\mathcal{A}$ is not minimal) $(\mathsf{literals}(\mathcal{A})\setminus\mathsf{base}(\mathcal{A}))\cap\Pi'=\{\langle\ell,t\rangle,\ldots\}\neq\varnothing$

($\mathcal{A}$ is not consistent) $(\mathsf{literals}(\mathcal{A})\setminus\mathsf{base}(\mathcal{A}))\cap\{\sim\langle\ell,t\rangle\}_{\langle\ell,t\rangle\in\Pi'}\neq\varnothing$

($\mathcal{A}$ is not defeasibly min.) there is $\mathcal{B}\subseteq\Pi'\cup A(\pi)$ s.t. $(\Pi'\cup A(\pi)\cup\mathsf{base}(\mathcal{B}),\Delta')$ is a program, and $\mathcal{B}$ is an argument in it with $\mathsf{concl}(\mathcal{B})\in\mathsf{literals}(\mathcal{A})\setminus\mathsf{base}(\mathcal{A})$, and finally $\mathcal{B}$ is not a sub-argument of $\mathcal{A}$

Then we resp. say that $\mathcal{B}=\{\langle\ell,t\rangle\}$ is a *strict threat* or that $\mathcal{B}$ is a *strict pre-threat* (in $\mathbb{M}'$). In the latter case, if moreover $\mathcal{B}$ is an argument in the program $\mathbb{P}'\oplus\pi$, then we also say that $\mathcal{B}$ is a *strict threat*. These strict (pre-)threats will be denoted $[\ldots,\mathcal{A},\mathcal{B}]\in\mathcal{T}^+_{\mathbb{P}\oplus\pi}(\mathcal{A})$ and, resp., $[\mathcal{A}_k,\ldots,\mathcal{A}_n,\mathcal{A},\mathcal{B}]\in\mathcal{T}^+_{\mathbb{P}\oplus\pi}(\mathcal{A}_k)$.

Definition 3.3.4 (Free interpretation)**.** Let $\mathbb{M}^{m+1}=(\mathbb{P}^{m+1},A^{m+1},G)$ be the planning domain of agent $f(m+1)$ at turn $m+1$. Assume either that an element $\pi\in\mathsf{Preplans}_m$ is a plan for $\mathbb{M}^{m+1}$, or that $(\pi,\pi^{(\mathbb{M}^m)^+})\in\mathsf{Plans}_m$. We define the *free interpretation of* π in $\mathbb{M}^{m+1}$, as the triple

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

$$\begin{array}{rcl}
A(\pi^{(\mathbb{M}^{m+1})^+}) & = & A(\pi^{\mathbb{M}^{m+1}})\\
\mathsf{Trees}(\pi^{(\mathbb{M}^{m+1})^+}) & = & \{\mathcal{T}^+_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k)\mid\mathcal{A}_k\text{ is an arg. step in }\pi\},\text{ with}\\
\mathcal{T}^+_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k) & = & \mathcal{T}^*_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k)\cup\\
 & & \cup\left\{\begin{array}{c}\Lambda^\cap[\mathcal{B}]\text{ is a (strict)}\\ \text{pre-threat for }\pi\end{array}\ :\ \begin{array}{c}\Lambda\in\mathcal{T}^*_{\mathbb{P}^{m+1}\oplus\pi}(\mathcal{A}_k)\\ \text{for some }\mathcal{A}_k\end{array}\right\}\\
 & & \cup\left\{\begin{array}{c}\Lambda^\cap[\mathcal{B},\mathcal{D}]\text{ is a}\\ \text{(strict) pre-threat}\end{array}\ :\ \begin{array}{c}\Lambda^\cap[\mathcal{B}]\text{ is a pre-}\\ \text{threat for }\pi\end{array}\right\}\\
\mathsf{goals}(\pi^{(\mathbb{M}^{m+1})^+}) & = & \mathsf{goals}(\pi^{\mathbb{M}^{m+1}})
\end{array}$$

Now we can formally define the messages communicated at each turn $m \geq 0$,

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

from an initially given multi-planner domain $\langle \mathbb{M}_a^0 \rangle_{a \in \mathsf{Ag}}$. As above, we let $\mathbb{M}^m$ denote the planning domain $\mathbb{M}_{f(m)}^m$. These communications are defined by simultaneous induction with the updated planning domains $\mathbb{M}_a^m$ of the agents. Finally, let us remark that if some set or tuple, say of the form X_m, is undefined, it must be read as the empty set $X_m = \varnothing$.

$$\begin{array}{lcl}
\mathsf{turn}(0) & = & (\ \varnothing,\ \varnothing,\ \{\pi_\varnothing\},\ \varnothing\ ) \\
\mathsf{turn}(m+1) & = & (\ \mathsf{Preplans}_{m+1},\ \mathsf{Plans}_{m+1},\ \mathsf{Trueplans}_{m+1},\ \mathsf{Data}_{m+1}\ )
\end{array}$$

$$\begin{array}{lcl}
\mathsf{Preplans}_{m+1} & = & (\mathsf{Preplans}_m \smallsetminus \mathsf{Preplans}_{m-|\mathsf{Ag}|}) \cup \\
 & & \cup \{\pi(\mathcal{A}) \text{ is a pre-plan for } \mathbb{M}_{f(m+1)}^{m+1} \mid \pi \in \mathsf{Trueplans}_m\} \\
\mathsf{Plans}_{m+1} & = & \left\{ (\pi, \pi^{(\mathbb{M}_{f(m+1)}^{m+1})^+}) \ : \begin{array}{c} (\pi, \pi^{(\mathbb{M}_{f(m)}^m)^+}) \in \mathsf{Plans}_m, \text{ or} \\ \pi \in \mathsf{Preplans}_{m+1} \\ \text{and } \pi \text{ is a plan for } \mathbb{M}_{f(m+1)}^{m+1} \end{array} \right\} \\
\mathsf{Trueplans}_{m+1} & = & \mathsf{Trueplans}_m \cup \\
 & & \cup \left\{ \begin{array}{c} \pi \in \mathsf{Plans}_{m+1} \text{ is a} \\ \text{plan for } \mathbb{M}_{f(m+1)}^{m+1} \end{array} : \begin{array}{c} (\pi, \pi^{(\mathbb{M}^m)^+}) \in \mathsf{Plans}_m \text{ and} \\ \pi^{(\mathbb{M}^{m+1})^+} = \pi^{(\mathbb{M}^{m+1-|\mathsf{Ag}|})^+} \end{array} \right\} \\
\mathsf{Data}_{m+1} & = & \mathsf{Data}_m \ \cup\ (\Pi^{m+1} \cap \mathsf{goals}(\pi^{(\mathbb{M}^m)^+})) \cup \\
 & & \cup \left\{ \mathsf{e} \in A^{m+1} \ : \begin{array}{c} \langle \mu_\mathsf{e}, t_\mathsf{e} \rangle \in \mathsf{base}(\mathcal{A}), \text{ for some} \\ \pi(\mathcal{A}) \in \mathsf{Preplans}_{m+1} \smallsetminus \mathsf{Preplans}_m \end{array} \right\}
\end{array}$$

$$\begin{array}{lcl}
\mathbb{M}_a^{m+1} & = & \begin{cases} \mathbb{M}_a^m & \text{if } a \neq f(m+2) \\ (\mathbb{P}_{f(m+2)}^{m+1}, A_{f(m+2)}^{m+1}, G) & \text{if } a = f(m+2) \end{cases} \\
 & & \text{where} \\
A_{f(m+2)}^{m+1} & = & A_{f(m+2)}^m \cup \{\mathsf{e} \in \mathsf{Data}_{m+1} \mid \mathsf{e} \text{ is an action } \} \\
\Pi_{f(m+2)}^{m+1} \cup \Delta_{f(m+2)}^{m+1} & = & \Pi_{f(m+2)}^m \ \cup\ \Delta_{f(m+2)}^m \ \cup\ \{\langle \ell, t \rangle \mid \langle \ell, t \rangle \in \mathsf{Data}_{m+1}\} \cup \\
\text{Note:} & & \left\{ \delta \in \begin{array}{c} \Pi_{f(m+1)}^{m+1} \\ \cup \\ \Delta_{f(m+1)}^{m+1} \end{array} : \begin{array}{c} \delta \in \mathcal{A} \smallsetminus \mathsf{base}(\mathcal{A}), \\ \text{for some } \pi(\mathcal{A}) \in \mathsf{Preplans}_{m+1}, \\ \text{or some } \Lambda^\cap[\mathcal{A}] \in \mathcal{T}^+_{\mathbb{P}_{f(m+1)}^{m+1} \oplus \pi}(\mathcal{A}_k) \\ \text{in a plan } \pi = \pi_\varnothing(\ldots, \mathcal{A}_k, \ldots) \\ \text{with } (\pi, \pi^{(\mathbb{M}_{f(m+1)}^m)^+}) \in \mathsf{Plans}_{m+1} \end{array} \right\} \\
\text{if } \delta \text{ has } \leftarrow, & & \\
\delta \in \Pi_{f(m+2)}^{m+1}; & & \\
\text{and if } \delta \text{ has } \prec, & & \\
\delta \in \Delta_{f(m+2)}^{m+1} & &
\end{array}$$

In summary, pre-plans π are generated within the sets $\mathsf{Preplans}_{(\ldots)}$. Eventually some of them, say at turn $m+1$, are seen as plans according to agent

$f(m+1)$, who moves them to the set Plans_{m+1} together with her free interpretation of these plans, denoted by pairs $(\pi, \pi^{(\mathbb{M}^{m+1})^+})$. Afterwards, agents search for threats against their plan steps or against the claim that this π actually defines a plan. If the latter fails to be shown, the plan (including all the threats detected) is communicated as a true plan, from where it can be further refined.

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

Algorithm 3: The algorithm for agent a in the dialogue-based planning method for t-DeLP backward planning.

In Example 3.3.5 (see below), a decentralized version of Example 2.3.9 is considered. In this new version, none of the agents start with full knowledge of the scenario, unlike the central planner from Example 2.3.9. See also Figure 3.3 for an illustration of the two-agent dialogue to solve Example 3.3.5. The plan steps and threats discussed in this dialogue are the same than in the previous Example 2.3.9. Below we repeat Figure 3.4 where these elements are depicted.

Example 3.3.5 (Table Lifting; cont'd)**.** We rewrite Example 2.3.9 in the form of a multi-planner problem with two agents $\mathsf{Ag} = \{a_1, a_2\}$. Now the agents also wish to lift the table without breaking the vase, but they ignore that the other agent can lift the other side of the table, and more importantly a_2 ignores the

rule δ_7 stating that *objects lying in non-horizontal surfaces tend to fall off*. The initial planning domain of each agent $\mathbb{M}_a = ((\Pi_a, \Delta_a, A_a, G)$ are defined in terms of that for the central planner from Ex. 2.3.9.

$$\begin{array}{rclcrcl} \Pi_{a_1} & = & \Pi_{\mathsf{Ag}} & & \Pi_{a_2} & = & \Pi_{\mathsf{Ag}} \\ \Delta_{a_1} & = & \Delta_{\mathsf{Ag}} & & \Delta_{a_2} & = & \Delta_{\mathsf{Ag}} \smallsetminus \{\delta_7\} \\ A_{a_1} & = & \{\mathsf{lift}.\mathsf{N}^t\}_{0\leq t<10} & & A_{a_2} & = & \{\mathsf{lift}.\mathsf{S}^t\}_{0\leq t<10} \end{array}$$
$$G \quad = \quad \{\langle l_{\mathsf{N}}, 10\rangle,\ \langle l_{\mathsf{S}}, 10\rangle,\ \langle \sim b, 10\rangle\}$$

3.4 Soundness and Completeness of the Dialogue-based Plan Search algorithm

Before proceeding with the proof for soundness and completeness of the dialogue-based algorithm, we observe some auxiliary results. First, it can be easily seen by induction that planning domains keep expanding during the dialogues, always with elements from other agents, hence from elements in the centralized planning domain $\mathbb{M}_{\mathsf{Ag}}$.

Fact 3.4.1. For each turn m and each $a \in \mathsf{Ag}$, we have $\mathbb{M}_a^m \sqsubseteq \mathbb{M}_a^{m+1}$ and $\mathbb{M}_a^m \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$.

Lemma 3.4.2. *Let $\pi = \pi_n$ be a plan for two planning domains $\mathbb{M}', \mathbb{M}$, both of the form $((\cdot,\cdot),\cdot, G)$ and with $\mathbb{M}' \sqsubseteq \mathbb{M}$, and moreover satisfying $\pi_k^{\mathbb{M}'} = \pi_k^{\mathbb{M}}$, for each $k \leq n$. Then, if $\pi(\mathcal{A})$ is again a plan for both $\mathbb{M}', \mathbb{M}$, we have*

$$A(\pi(\mathcal{A})^{\mathbb{M}'}) = A(\pi(\mathcal{A})^{\mathbb{M}}) \qquad \textit{and} \qquad \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}}) = \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}'}) \smallsetminus \Pi$$

Proof. For the identity on actions, let $A^* \subseteq A' \subseteq A$ be a $\subseteq$-minimal set of actions supporting $\mathcal{A}$ in $\mathbb{M}'$, that is, satisfying

$$\mathsf{post}[A(\pi^{\mathbb{M}'}) \cup A^*] \cup \Pi' \supseteq \mathsf{base}(\mathcal{A})$$

Then, notice that

(1) A^* also supports $\mathcal{A}$ in $\mathbb{M}$, that is, $\mathsf{post}[A(\pi^{\mathbb{M}}) \cup A^*] \cup \Pi \supseteq \mathsf{base}(\mathcal{A})$. The reason is that $A(\pi^{\mathbb{M}'}) = A(\pi^{\mathbb{M}})$ and $\Pi \supseteq \Pi'$.

(2) A^* is also $\subseteq$-minimal w.r.t. (1), since $\Pi \smallsetminus \Pi'$ does not contain literals of the form $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$, in particular, those occurring in $\mathsf{base}(\mathcal{A})$.

For the identity claim on goals, consider first the case where $\mathcal{A}$ is an argument step. If we let $\pi = \pi_k = \pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_k)$, then note that the assumption $\pi^{\mathbb{M}'} = \pi^{\mathbb{M}}$ implies the identity between previous goals occurring in initial fragments π_i. Then, we have

$$
\begin{aligned}
& \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}}) \\
= \quad & (\mathsf{goals}(\pi^{\mathbb{M}}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi \cup \mathsf{OldGoals}(\pi^{\mathbb{M}}))) && \text{(by Def. 2.5.3)} \\
= \quad & (\mathsf{goals}(\pi^{\mathbb{M}'}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi \cup \mathsf{OldGoals}(\pi^{\mathbb{M}}))) \\
& && \text{(since } \pi^{\mathbb{M}} = \pi^{\mathbb{M}'} \text{ implies } \mathsf{goals}(\pi^{\mathbb{M}}) = \mathsf{goals}(\pi^{\mathbb{M}'})) \\
= \quad & (\mathsf{goals}(\pi^{\mathbb{M}'}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi \cup \mathsf{OldGoals}(\pi^{\mathbb{M}'}))) \\
& \text{(since } {\pi_0}^{\mathbb{M}} = {\pi_0}^{\mathbb{M}'}, \ldots, {\pi_n}^{\mathbb{M}} = {\pi_n}^{\mathbb{M}'} \text{ implies } \mathsf{OldGoals}(\pi^{\mathbb{M}}) = \mathsf{OldGoals}(\pi^{\mathbb{M}'})) \\
= \quad & (\mathsf{goals}(\pi^{\mathbb{M}'}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi' \cup \Pi \cup \mathsf{OldGoals}(\pi^{\mathbb{M}'}))) \\
& && \text{(since } \Pi' \subseteq \Pi \text{ implies } \Pi' \cup \Pi = \Pi) \\
= \quad & ((\mathsf{goals}(\pi^{\mathbb{M}'}) \cup \mathsf{pre}[A^*]) \setminus (\{\langle \ell, t\rangle\} \cup \Pi' \cup \mathsf{OldGoals}(\pi^{\mathbb{M}'}))) \setminus \Pi \\
= \quad & \mathsf{goals}(\pi(\mathcal{A})^{\mathbb{M}'}) \setminus \Pi && \text{(by Def. 2.5.3)}
\end{aligned}
$$

For the case where $\mathcal{A}$ is a threat resolution move, the proof analogous: just delete the set $\{\langle \ell, t\rangle\} \cup \ldots$ everywhere, and replace Def. 2.5.3 by Def. 2.5.5. $\square$

Theorem 3.4.3 (Soundness)**.** *Let π be the output of the dialogue-based plan search algorithm for some given multi-planner domain $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$. Then π is a solution for $\mathbb{M}_{\mathsf{Ag}}$*

Proof. Let π_n denote the output sequence $\pi_n = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ and as usual let π_k denote its initial fragment $\pi_k = \pi_\varnothing(\mathcal{A}_1, \ldots, \mathcal{A}_k)$. Moreover, let $m_0 < \ldots < m_n$ be a sequence of turns satisfying $\pi_k \in \mathsf{Trueplans}_{m_k}$, and with each m_k minimal with this property. These turns m_k clearly exist for each π_k with $0 \leq k \leq n$, as can be seen by inspection of the definitions for $\mathsf{Trueplans}_{(\cdot)}$ and the other sets $\mathsf{Plans}_{(\cdot)}, \mathsf{Preplans}_{(\cdot)}$.

The proof is by induction on the length k (of the initial fragments π_k). For each planning domain $\mathbb{M}$ satisfying $\mathbb{M}^{m_k} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$ we show the next claims:

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

(Base Case $k = 0$) Note that $m_0 = 0$. Now, claim (1) is obvious, since $\pi_0 = {\pi_\varnothing}^{\mathbb{M}} = (\varnothing, \varnothing, G)$ is a plan for any planning domain $\mathbb{M}$ of the form $\mathbb{M} = ((\cdot, \cdot), \cdot, G)$; this includes all the planning domains generated in the dialogue and more generally any of the planning domains $\mathbb{M}$ satisfying $\mathbb{M}_a^0 \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. Claim (2), follows from the fact that each interpretation of plan π_0 is $(\varnothing, \varnothing, G)$ among all these planning domains.

(Ind. Case $k \Rightarrow k+1$) Assume (Ind. Hyp.) that (1) and (2) hold for $\pi_0, \ldots, \pi_k$, and arbitrary domains $\mathbb{M}$ such that $\mathbb{M}^{m_k} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. We show that (1) and (2) hold for the plan $\pi_{k+1} \in \mathsf{Trueplans}_{m_{k+1}}$ and any planning domain $\mathbb{M}$ satisfying $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. Let then $\mathbb{M}$ be an arbitrary planning of this form.

Claim (1). Note that $m_k < m_{k+1}$ implies $\mathbb{M}^{m_k} \sqsubseteq \mathbb{M}^{m_{k+1}}$, so by the Ind. Hyp. we have in particular that π_k is a plan for $\mathbb{M}$. The proof that π_{k+1} is also a plan for $\mathbb{M}$ is by cases, depending on which type of plan step $\mathcal{A}_{k+1}$ is. In fact, we only show the case where $\mathcal{A}_{k+1}$ is an argument step, since the proof for the case

of a threat resolution move $\Lambda = [\mathcal{A}_i, \ldots, \mathcal{B}, \mathcal{A}_{k+1}]$ is entirely similar. (Below we point out the necessary adjustments for the latter proof.)

(Case $\mathcal{A}_{k+1}$ is an argument step) We check that $\pi_k(\mathcal{A}_{k+1})$ and $\mathbb{M}$ satisfy Definition 2.5.3. Let m' with $m_k < m' < m_{k+1}$ be the turn where π_{k+1} first occurs in $\mathsf{Plans}_{m'}$ (clearly, this m' exists by definition of $\mathsf{Trueplans}_{m+1}$, etc.). Moreover, π_{k+1} is a plan for $\mathbb{M}^{m'}$ (i.e. it satisfies Def. 2.5.3), given by the elements: $\mathcal{A}^-_{k+1} = \mathcal{A}_{k+1} \setminus \mathsf{base}(\mathcal{A}_{k+1}) \subseteq \Pi^{m'}_{f(m')} \cup \Delta^{m'}_{f(m')}$; and some appropriate set $A^* \subseteq A^{m'}$.

First, we need to check that $\mathsf{concl}(\mathcal{A}_{k+1})$ is an open goal in the plan $\pi_k^{\mathbb{M}}$. By the Ind. Hyp. on claim (2), we have that the goal $\langle \ell, t\rangle = \mathsf{concl}(\mathcal{A}_{k+1}) \in \mathsf{goals}(\pi_k)^{\mathbb{M}^{m_k}}$ is also in $\mathsf{goals}(\pi_k)^{\mathbb{M}^{m_{k+1}}}$. To check that this goal is also in $\mathsf{goals}(\pi_k)^{\mathbb{M}}$, assume the contrary: $\langle \ell, t\rangle \notin \mathsf{goals}(\pi_k)^{\mathbb{M}}$. Using Lemma 3.4.2, the last two claims imply that $\langle \ell, t\rangle \in \Pi$. But since $\Pi \subseteq \Pi_{\mathsf{Ag}} = \bigcup_{a\in\mathsf{Ag}} \Pi^0_a$, we obtain that $\langle \ell, t\rangle \in \Pi^0_a$ for some $a \in \mathsf{Ag}$. Now, the assumption $\pi_{k+1} \in \mathsf{Trueplans}_{m+1}$ implies that

$$\pi_{k+1}^{(\mathbb{M}^{m_{k+1}-|\mathsf{Ag}|})^+} \text{ is defined, so } \begin{cases} (\pi_{k+1}, \pi_{k+1}^{(\mathbb{M}^{m_{k+1}-|\mathsf{Ag}|})^+}) \in \mathsf{Plans}_{m_{k+1}-|\mathsf{Ag}|} \\ \qquad \vdots \\ (\pi_{k+1}, \pi_{k+1}^{(\mathbb{M}^{m_{k+1}})^+}) \in \mathsf{Plans}_{m_{k+1}} \end{cases}$$

In particular, $(\pi_{k+1}, \pi_{k+1}^{(\mathbb{M}^{m''})^+}) \in \mathsf{Plans}_{m''}$ for some such $m_{k+1} - |\mathsf{Ag}| \leq m'' \leq m_{k+1}$ satisfying $f(m'') = a$. The latter implies that $\mathsf{Data}_{m''}$, $\langle \ell, t\rangle \in \mathsf{Data}_{m''} \subseteq \mathsf{Data}_{m_{k+1}-1}$, so finally $\{\langle \ell, t\rangle \mid \langle \ell, t\rangle \in \mathsf{Data}_{m_{k+1}-1}\} \subseteq \Pi^{m_{k+1}}$, contradiction.

Second, not only π_k is a plan for $\mathbb{M}$, but also the above elements $\mathcal{A}^-_{k+1}$ and A^* exist in $\mathbb{M}$. The latter is seen by $\mathbb{M}^{m'} \sqsubseteq \mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M}$.

Finally, it only remains to check that conditions (i)-(v) from Def. 2.5.3 also hold for the planning domain $\mathbb{M}^{m_{k+1}}$ and the same elements π_k, $\mathcal{A}^-_{k+1}$ and A^*.

(i) $A(\pi_k{}^{\mathbb{M}}) \cup A^*$ *is non-overlapping*, since by the Ind. Hyp. on (2), this set is identical to $A(\pi_k{}^{\mathbb{M}^{m'}_{f(m')}}) \cup A^*$, which is non-overlapping for each $a \in \mathsf{Ag}$.

(ii) $(\mathbb{P} \oplus \pi_k) \oplus A^*$ *is a t-DeLP program*; this follows from the fact that $\Pi \subseteq \Pi_{\mathsf{Ag}}$ and the latter is consistent, so Π must be consistent as well. The addition of arbitrary elements $\langle \mu_{\mathsf{e}}, t_{\mathsf{e}}\rangle$ from actions preserves this consistency.

(iii) $\mathcal{A}_{k+1}$ *is an argument for* $\langle \ell, t\rangle$ *in this program*; that $\mathcal{A}_{k+1} \subseteq ((\Pi \oplus \pi_k) \oplus A^*) \cup \Delta$ was essentially shown above. We check the conditions (1)-(4) from Def. 1.3.5:

(1) $(\mathcal{A}_{k+1} \cap \Delta) \cup (\Pi \oplus \pi_k) \vdash \langle \ell, t\rangle$; this is preserved from $\mathbb{M}^{m'}$, where it holds by assumption, and the fact $\mathbb{M}^{m'} \sqsubseteq \mathbb{M}$;

(2) $\Pi \cup (\mathcal{A}_{k+1} \cap \Delta)$ *is consistent*; for this, assume the contrary, that is, some $\langle \ell', t'\rangle$ exists in $\sim\mathsf{literals}(\mathcal{A}_{k+1}) \cap \mathrm{Cn}(\Pi)$. Let then $\mathcal{B}$ be a

strict argument for this $\langle \ell', t' \rangle$. Clearly, $\mathcal{B}$ must consist of a strict fact $\langle \ell_0, t_0 \rangle$, and possibly a unique mutex rule δ. Consider the former case. Since $\langle \ell_0, t_0 \rangle \in \Pi \subseteq \Pi_{\mathsf{Ag}}$, let $j < |\mathsf{Ag}|$ be minimal with the property $\langle \ell_0, t_0 \rangle \in \Pi_{f(m'+j)}$. Then $[\mathcal{A}_{k+1}, \mathcal{B}] \in \mathcal{T}^{+}_{\mathbb{P}^{m'+j} \oplus \pi_{k+1}}(\mathcal{A}_{k+1})$. Clearly, $m' + j < m_{k+1}$, so $\mathcal{B}$ is a strict argument in $\mathbb{P}^{m_{k+1}} \oplus \pi_{k+1}$, and hence π_{k+1} is not a plan for $\mathbb{M}_{k+1}$, contradicting the assumption $\pi_{k+1} \in \mathsf{Trueplans}_{m_{k+1}}$. The proof for the case $\mathcal{B} = \{\langle \ell_0, t_0 \rangle, \delta\}$ for some mutex rule δ is similar, now $\mathbb{P}^{m'+j}$ containing δ, and defining some j' with $j \leq j'' \leq |\mathsf{Ag}|$ such that $\mathcal{B}$ is an argument in $\mathbb{P}^{m'+j'}$.

(3) $\mathcal{A}_{k+1} \cap \Delta$ *is* $\subseteq$*-minimal w.r.t.* (1)-(2); for (1), the proof is similar to the previous of (2), but instead of building a strict argument for the negation of some (defeasibly derived) literal in $\mathcal{A}_{k+1}$ we build the strict argument for one of these literals. For (2), the $\subseteq$-minimality of $\mathcal{A}_{k+1}$ is obvious.

(4) $\mathcal{A}_{k+1} \cap \Pi$ *is* $\subseteq$*-minimal satisfying* $(\mathcal{A}_{k+1} \cap \Delta) \cup (\mathcal{A}_{k+1} \cap \Pi)$; Again the proof is similar, the strict argument now is an argument for some of the strictly derived literals within $\mathcal{A}_{k+1}$.

(iv) A^* is $\subseteq$*-minimal w.r.t.* (iii). By definition of A^* in $\mathbb{M}^{m'}$.

(v) this *consistency condition* from Def. 2.5.3 is the exactly the same between $\mathbb{M}^{m'}$ (where it holds by assumption) and the arbitrarily selected $\mathbb{M}$ (note that the two sets of previous goals are the same because of the Ind. Hyp. for (2)).

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

For *actions*, note that the Ind. Hyp. for (2) implies $A({\pi_k}^{\mathbb{M}^k}) = A({\pi_k}^{\mathbb{M}^{k+1}}) = A({\pi_k}^{\mathbb{M}}) = A({\pi_k}^{\mathbb{M}_{\mathsf{Ag}}})$, since $\mathbb{M}^k \sqsubseteq \mathbb{M}^{k+1} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. On the other hand, by the previous claim (1) in the inductive case, we know that π_{k+1} is a plan for arbitrary $\mathbb{M}$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. The latter two facts jointly permit to apply Lemma 3.4.2 and conclude that $A({\pi_{k+1}}^{\mathbb{M}^{m_{k+1}}}) = A({\pi_{k+1}}^{\mathbb{M}}) = A({\pi_{k+1}}^{\mathbb{M}_{\mathsf{Ag}}})$, for arbitrary $\mathbb{M}$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$.

For *goals*, we again apply Lemma 3.4.2 to first obtain that $\mathsf{goals}({\pi_{k+1}}^{\mathbb{M}}) = \mathsf{goals}({\pi_{k+1}}^{\mathbb{M}}) \setminus \Pi$, so if this set is not the same that $\mathsf{goals}({\pi_{k+1}}^{\mathbb{M}^{m_{k+1}}})$, then the latter contains some $\langle \ell', t \rangle \in \Pi \setminus \Pi^{m_{k+1}}$. Using $\Pi \subseteq \Pi_{\mathsf{Ag}}$, we can reason similarly to the above proof that $\mathsf{concl}(\mathcal{A}_{k+1}) \in \mathsf{goals}({\pi_{k+1}}^{\mathbb{M}})$ in claim (1), and reach a contradiction: $\langle \ell', t' \rangle \in \mathsf{Data}_{m_{k+1}-1} \subseteq \Pi^{m_{k+1}}$.

For *sub-trees* $\mathcal{T}^{*}_{\mathbb{P} \oplus \pi_{k+1}}(\mathcal{A})$. Note first that the set of arguments $\mathcal{A}$ for which these sub-trees are defined are the same than those existing in $\mathsf{Trees}({\pi_{k+1}}^{\mathbb{M}^{m'}})$. And the latter trees are defined for the same than those sub-trees $\mathcal{T}^{+}_{\mathbb{P}^{m'+j} \oplus \pi_{k+1}}(\mathcal{A})$, for any $j \leq m_{k+1} - m'$. Let then $\mathcal{A}$ be an arbitrary argument step.

(Sub-Case $\mathcal{A}_{k+1} \neq \mathcal{A}$) Note first that the argumentation lines of the form $[\mathcal{A}, \ldots, \mathcal{B}]$ are the same between the full trees $\mathcal{T}_{{\mathbb{P}^{m_{k+1}} \oplus \pi_{k+1}}^{\mathbb{M}^{m_{k+1}}}}(\mathcal{A})$ and

$\mathcal{T}_{\mathbb{P}\oplus\pi_{k+1}{}^{\mathbb{M}}}(\mathcal{A})$, for any $\mathbb{M}$ with $\mathbb{M}^{m_{k+1}} \sqsubseteq \mathbb{M} \sqsubseteq \mathbb{M}_{\mathsf{Ag}}$. To see this, assume the contrary.

First, suppose that some threat $[\mathcal{A}, \ldots, \mathcal{B}]$ exists in ${\pi_{k+1}}^{\mathbb{M}^{m_{k+1}}}$ but not in ${\pi_{k+1}}^{\mathbb{M}}$. Clearly, this $\mathcal{B}$ is not a strict argument, since $\mathcal{A}$ exists in $\mathbb{P}^{m_{k+1}}$. The absence of this threat $[\mathcal{A}, \mathcal{B}]$ in ${\pi_{k+1}}^{\mathbb{M}}$ can only mean that $\mathcal{B}$ is not an argument in $\mathbb{P} \oplus \pi_{k+1}$. Since $\mathbb{P}^{m_{k+1}}$ is piecewise included in $\mathbb{P}$, and the derivability of $\mathcal{B}$ is obviously preserved, one of the conditions (2)-(4) from Def. 1.3.5 must fail. This can only mean that a strict argument $\mathcal{C}$ exists either with $\sim\mathsf{concl}(\mathcal{C}) \in \mathsf{literals}(\mathcal{B})$, or with $\mathsf{concl}(\mathcal{C}) \in \mathsf{literals}(\mathcal{B})$. In either case, this $\mathcal{C}$ is a strict pre-threat for $\mathcal{B}$. Moreover, it can be shown as usual that $\mathcal{B}$ will have been built before m_{k+1}, so $[\mathcal{A}, \mathcal{B}, \mathcal{C}]\mathcal{T}^{+}_{\mathbb{P}^{m'+j}}(\mathcal{A})$, for some $m' + j < m_{k+1}$, so by def. of $\mathbb{M}^{m_{k+1}}$, $\mathcal{C}$ must exist in $\mathbb{P}^{m_{k+1}} \oplus \pi_{k+1}$, thus contradiction the assumption that $[\mathcal{A}, \mathcal{B}]$ is a threat in ${\pi_{k+1}}^{\mathbb{M}^{m_{k+1}}}$.

Second, suppose that a threat $[\mathcal{A}, \mathcal{B}]$ exists in ${\pi_{k+1}}^{\mathbb{M}}$ but not in ${\pi_{k+1}}^{\mathbb{M}^{m_{k+1}}}$. We show the latter is impossible. Define $\mathcal{B}_a = \mathcal{B} \cap ((\Pi^0_a \oplus \pi_{k+1}) \cup \Delta^0_a)$, and for each $0 \leq j < m_{k+1} - m'$ inductively define

$$\mathcal{B}^j = \quad \text{the largest terminal fragment of } \mathcal{B} \text{ with } \mathcal{B}^j \subseteq \mathcal{B}_{f(m'+j)} \cup \textstyle\bigcup_{j'<j} \mathcal{B}^{j'}$$

It is routine to check, for each such j, that $\mathcal{B}^j \subseteq (\Pi^{m'+j} \oplus \pi_{k+1}) \cup \Delta^{m'+j}$, so $[\mathcal{A}_{k+1}, \mathcal{B}^j] \in \mathcal{T}^{+}_{\mathbb{P}^{m'+j}\oplus\pi_{k+1}}(\mathcal{A}_{k+1})$; in addition, it can be seen that ${\pi_{k+1}}^{(\mathbb{M}^{m'+j})^+} \neq {\pi_{k+1}}^{(\mathbb{M}^{m'+j-|\mathsf{Ag}|})}$ –using an argument similar to the above for the claim $\mathsf{concl}(\mathcal{A}_{k+1}) \in \mathsf{goals}({\pi_k}^{\mathbb{M}})$. Thus, $\mathcal{B}$ is a strict argument in $\mathbb{P}^{m_{k+1}} \oplus \pi_{k+1}$, and hence π_{k+1} is not a plan for $\mathbb{M}_{k+1}$, contradicting the assumption that $\pi_{k+1} \in \mathsf{Trueplans}_{m_{k+1}}$.

(Sub-Case $\mathcal{A}_{k+1} = \mathcal{A}$.) This case is the same as above, except that threats are now of the form $[\mathcal{A}, \mathcal{B}]$. In addition, though, we must also rule out the existence of a strict pre-threat $\mathcal{B}$ against $\mathcal{A}$ itself. But again, the proof of this is as usual, showing that such $\mathcal{B}$ would have been built before the turn m_{k+1}, in the corresponding $\mathcal{T}^+$-trees.

(Case $\mathcal{A}_{k+1}$ is a threat resolution move) As we mentioned, the proof for this case is analogous to the former case, but now with a plan step of the form $[\mathcal{A}_i, \ldots, \mathcal{B}, \mathcal{A}_{k+1}]$. The major change is that we use the Ind. Hyp. for the identity $\mathsf{Trees}({\pi_k}^{\mathbb{M}^{m_{k+1}}}) = \mathsf{Trees}({\pi_k}^{\mathbb{M}})$, rather than for $\mathsf{goals}({\pi_k}^{\mathbb{M}^{m_{k+1}}}) = \mathsf{goals}({\pi_k}^{\mathbb{M}})$, to show that $[\mathcal{A}_i, \ldots, \mathcal{B}]$ exists, so $[\mathcal{A}_i, \ldots, \mathcal{B}, \mathcal{A}_{k+1}]$ is a plan step for π_k.

This concludes the inductive proof for claims (1) and (2). We complete the proof for the theorem using these claims.

As a particular case of (1)-(2), π_n is a plan for $\mathbb{M}_{\mathsf{Ag}}$. To see that this plan π is a *solution* for $\mathbb{M}_{\mathsf{Ag}}$, recall that π satisfies the Terminating Condition for $\mathbb{M}^{m_n}$.

Hence, on the one hand we have that $\mathsf{goals}(\pi^{\mathbb{M}^{m_n}}) = \varnothing$. This, together with Lemma 3.4.2 implies that

$$\mathsf{goals}(\pi^{\mathbb{M}_{\mathsf{Ag}}}) = \mathsf{goals}(\pi^{\mathbb{M}_n}) \smallsetminus \mathbb{M}_{\mathsf{Ag}} = \varnothing \smallsetminus \mathbb{M}_{\mathsf{Ag}} = \varnothing$$

On the other hand,

$$
\begin{aligned}
& \mathsf{threats}(\pi^{\mathbb{M}_{\mathsf{Ag}}}) \\
= \;& \{\Lambda \in \mathcal{T}^{*}_{\mathbb{P}_{\mathsf{Ag}} \oplus \pi_n}(\mathcal{A}_k) \mid 1 \leq k \leq n \text{ and } \mathcal{A}_k \text{ is an arg. step}\} \\
= \;& \{\Lambda \in \mathcal{T}^{*}_{\mathbb{P}^{m_n} \oplus \pi_n}(\mathcal{A}_k) \mid 1 \leq k \leq n \text{ and } \mathcal{A}_k \text{ is an arg. step}\} \\
& \text{(by the above claim (2))} \\
= \;& \mathsf{threats}(\pi^{\mathbb{M}^{m_n}}) \\
= \;& \varnothing \text{ (since } \pi \text{ satisfies the Terminating Condition in } \mathbb{M}^{m_n})
\end{aligned}
$$

Hence we conclude that π_n is a plan for the planning domain $\mathbb{M}_{\mathsf{Ag}}$, and moreover that it satisfies the Terminating Condition. Now, using the proof of the Soundness Theorem 2.7.5 we conclude that the output π is a solution for $\mathbb{M}_{\mathsf{Ag}}$. □

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

Corollary 3.4.4. *Let $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$ be a multi-planner domain, and let π_n be the output of the dialogue-based algorithm for this $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$. Assume that $\pi_n \in \mathsf{Trueplans}_{m_n}$ with m_n minimal with this property. Then*

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

Finally, we conclude the study of the dialogue-based algorithm by showing that it is complete.

Theorem 3.4.5 (Completeness)**.** *Let $\langle \mathbb{M}_a \rangle_{a \in \mathsf{Ag}}$ be a multi-planner domain. If a solution A' exists for the centralized domain $\mathbb{M}_{\mathsf{Ag}}$, then the dialogue-based algorithm terminates with an output.*

Proof. From the assumption that a $\subseteq$-minimal solution A' exists, i.e. $G \subseteq \mathsf{warr}(\mathbb{P}_{\mathsf{Ag}} \diamond A')$, we first proceed as in the proof of the Completeness Theorem 2.8.1 (now for $\mathbb{M}_{\mathsf{Ag}}$). Thus, from the set of actions $A^{\star}$, we obtain the sets $\mathsf{Lines}, \mathsf{Steps}, \mathsf{Threats}$, and also a sequence $\pi_{\varnothing}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ where $\mathsf{Steps} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. Using Theorem 2.8.1, we know that $A({\pi_n}^{\mathbb{M}_{\mathsf{Ag}}}) = A'$ and $\mathsf{goals}({\pi^n}^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$ and $\mathsf{threats}({\pi_n}^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$.

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

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

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

Claim (1) can be shown analogously to the proof for the construction of pre-threats in Theorem 3.4.3. Claim (2) is a combination of proofs similar to those for the construction of pre-threats and open goals in Theorem 3.4.3.

It only remains to check that π_n satisfies the Terminating Condition for $\mathbb{M}_{m_n}$, but this follows from the above claims $\mathsf{goals}(\pi_n^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$ and $\mathsf{threats}(\pi_n^{\mathbb{M}_{\mathsf{Ag}}}) = \varnothing$ and the facts that

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

These two facts are shown as in Theorem 3.4.3. □

3.5 Conclusions and Related Work

In this chapter, we extended the centralized planning algorithms for multiple agents from Chapter 2 to planning by multiple agents. The soundness and completeness theorems for the proposed dialogue-based planning algorithm show, informally, that the dialogue terminates with agents agreeing upon some joint solution plan. In summary, the novelty of the present approach is the combination of: temporal reasoning, temporal planning, and decentralized planning for multi-agent systems.

Notice that these results involved informal notions of group or common knowledge, which cannot be formally studied within t-DeLP. These concepts, in contrast, have been systematically studied in the area of dynamic epistemic logic. In particular, in Part II of this thesis some of these studies are extended to planning problems involving these epistemic notions.

Multi-agent collaborative planning has been a topic of recent interest within the areas of planning, multi-agent systems and argumentation. The literature on planning (and more generally search) has been studying distributed versions of the corresponding algorithms for standard planning systems, e.g. [82].

All these distributed versions assume a collaborative approach, as we did for the present chapter. Most proposals divide into those addressing the problem of coordination after planning and those addressing coordination during the planning phase (see also [34] for the problem of coordination before planning). The present approach belongs to the second class where coordination of agents' actions, beliefs and plans takes place during the construction of plans.

Multi-agent argumentation is also related to the present framework. Among the argumentation tools used to solve single-agent planning or practical reasoning problems, some are based Dung's abstract argumentation [52].This has been used for reasoning about conflicting plans and generate consistent sets of goals [5, 79]. Further extensions of these works distinguish between belief arguments and goals arguments and include methods for comparing arguments based on the worth of goals and the cost of resources [124]. In any case, none of these works apply to a multi-agent environment. The work in [21] presents a dialogue based on an argumentation process to reach agreements on plan proposals. Unlike our focus on an argumentation-based construction of plans, this latter work is aimed at handling the interdependencies between agents' plans. Also related to the present work are the studies in multi-agent argumentation based on dialogues. See [126] for a complete review, and [135]for the particular case of DeLP.

This paper studies dialogues for distributed query answering problems in DeLP. Let us remark that in general multi-agent argumentation is a particular case of multi-agent planning (based on argumentation). The t-DeLP version of problems in dialogue-based query-answering in a logic program [135], for example, can be seen as problems of distributed planning where the initial state is the logic program, the query is the goal and the set of actions is empty.

In addition, logical approaches to the problem of multi-agent planning and communication can also be found in the literature. Among them, [73] addresses the present problem in a belief-desire-intention modal logic. On the other hand, theories in first-order logic have also been suggested to model scenarios with collaborative planning [48]; this work models the actions of communication and planning in the style of the Situation Calculus.

Finally, several related proposals, instead, address the most general case of multi-agent planning, where conflicting interests might exist. For this class of problems, called adversarial planning, the tools presented here are clearly insufficient: dialogues might in the general case involve strategically-minded agents, which need not be fully cooperative in communication (they might lie or simply not share relevant information). Most of the contributions addressing this kind of multi-agent planning problems adopt a game-theoretic approach. This is the case, for example, for multi-agent classical planning [89], and multi-agent STRIPS [29].

The present work is closely related to [110], [111] and [104].

Table 2. Dialogue corresponding to Example 3.3.5.

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

Figure 3.3: Dialogue for a decentralized version of Example 2.3.9. The notation used for (pre-)plans is, e.g., as follows: π_{mnk} denotes a plan $\pi_\varnothing(\mathcal{A}_m, \mathcal{A}_n, \mathcal{A}_k)$.

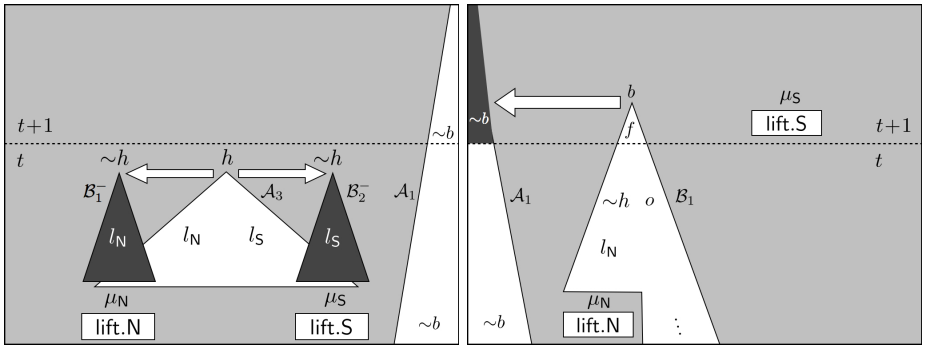


Figure 3.4: A representation of Example 2.3.9. (Left) A solution plan based on two simultaneous lifting actions at the interval $[t-1, t]$. (Right) A failed plan where agents non-simultaneously lift the table.

Part II

Planning in Dynamic Epistemic Logics

Introduction

The second part of this thesis is devoted to the study of backward planning in Dynamic Epistemic Logics, and in particular to a family of these logics, called Logics of Communication and Change (LCC). These logics contain dynamic modalities for agents' actions and epistemic modalities for agents' beliefs. The combination of these two types of modalities permits to reason about the effects of actions both upon the world and the minds of the agents. For example, a physical action like *closing a door* has physical effects *the door is closed* and epistemic effects *this agent knows that the door is closed*. The latter effects depend on the epistemic opportunities of the agents and their sensing capacities. This epistemic dimension is even more important in purely epistemic actions, traditionally not considered in the literature on planning. Purely epistemic actions include communicative actions between agents, or these agents' sensing actions. These actions have in common that, while their physical effects are negligible, they can have important consequences upon the behavior of self-motivated agents through changes in their belief systems.

In contrast to Part I, then, the present logics already contain all the elements of the (corresponding) state transition systems. Precisely, the Kripke semantics of any of these logics is indeed such an (epistemic) state transition system. It is also noteworthy that each such logic axiomatically characterizes the behavior of all the actions existing in the language of this logic.

For this reason, it seems natural to extend the study of dynamic epistemic logics into that of planning systems based on these logics. In these planning systems, the planner agent can pursue epistemic goals, e.g. *to learn something that another agent might know*, or she can pursue traditional physical goals, e.g. *to obtain someone's wallet* but in a way that takes agents' beliefs seriously, e.g. *to obtain someone's wallet without their knowledge*. In any case, the planner agent is able to decide what to say, where to look upon, and what to do in order to satisfy these goals. The scenarios that can be addressed by such a planner agent can be considerably complex in terms of social interactions.

Chapter 4 briefly reviews some dynamic epistemic logics in the literature, and presents the Logics of Communication and Change [139] with some detail. Then in Chapter 5, we study first algorithms for a planning system based on the Logics of Communication and Change. The planning system that immediately results from an LCC logic is deterministic, since so are the atomic actions definable in this family of logics. A Breadth First Search algorithm for plan search is studied and shown to be sound and complete.

In order to obtain a non-deterministic planning system, in Chapter 6 we study first an extension of LCC logics with choice and composition of actions. Finally, a Breadth First Search algorithm for strong non-deterministic planning is proposed in Chapter 7. This algorithm is shown to be sound and complete.

Chapter 4

Logics of Communication and Change

Dynamic Epistemic Logic is a recent area of interest in logics of multi-agent systems, focusing on the notions of action and belief, and the interactions between these two notions. Logics for agents with epistemic and communicative abilities have been developed in the last decades, ranging from epistemic logic (for individual, group or common belief or knowledge), to logics of announcements (public or private, truthful or lying), and finally incorporating ontic actions (i.e. physical or world-changing actions). All these logics have been unified within the single framework of Logics for Communication and Change [139].

After reviewing examples in epistemic logic and the logic of public announcements, the chapter describes the Logics of Communication and Change LCC, introduced by van Benthem, van Eijck and Kooi in [139]. This is a family of expressive dynamic epistemic logics capturing most of the previous work under a general model for actions. The action model for any such LCC logic can describe how the actual execution of an action is perceived by the agents, much like epistemic logics describe how the actual world is perceived by the different agents. The present chapter describes in detail the semantics and also the syntactic tools for the logics LCC. These tools will be later used in Chapter 6 to extend the LCC logics with program constructors, and in Chapters 5 and 7 to define plan search algorithms for planning. In this work, a general (translation-based) method provides a complete axiomatization.

Structure of the Chapter

In Section 4.1 we motivate this chapter by informally reviewing some semantics and examples from the literature on Epistemic Logic and Dynamic Epistemic Logic. After this, we proceed with a brief description of the Logics of Communication and Change from [139], including the definitions and basic results used in later chapters. This includes Section 4.2, where an epistemic reading of PDL

is considered, later used as the static base for LCC logics. Then, action models are introduced in Section 4.3. Finally, in Section 4.4, we present the syntactic tools and the reduction axioms that show the soundness and completeness of the LCC logics. We conclude this chapter with a list of some related work in dynamic epistemic logic, in Section 4.5. Except for the introduction in Section 4.1 and Lemma 4.4.3, this chapter entirely based on [139].

4.1 Introduction

Logic can be described as the study of valid inference in a language of propositions. Or, put negatively, a logic studies what propositions are compatible with some set of propositions in the language, e.g. a knowledge base of some agent. In the case of (multi-agent) epistemic Logic EL, both the agent and its knowledge base are made explicit and distinguished from (an external perspective on) the scenario or world. The above negative description of logic in terms of compatible propositions or possibilities, represents the agents' ignorance on the corresponding issues. Thus, one can reason about the epistemic possibilities still open to an agent, about other agents (their own epistemic possibilities) or about the world facts.

Epistemic Logic

Epistemic logic EL [76] is a formal study of the notions of knowledge and belief. A standard framework in the study of epistemic (or doxastic) logics is that of modal logic [40].

In (dynamic) epistemic logic, the language contains terms for each agent a in a finite set Ag, in addition to the usual set of atoms Var. These terms are used, in epistemic logic, as modalities $[a]\varphi$ expressing that *agent a knows that φ*, (again, one modal operator $[a]$ exists for each agent $a \in \mathsf{Ag}$). These logics can be extended with modalities for common knowledge $[C]$ or relativized common knowledge $[C^{\varphi}]$, though we will introduce them later.

Epistemic logics are used to model snapshots of epistemic notions (knowledge, belief) at a given moment; see Fig. 4.1 (a)–(c) for examples of epistemic snapshots.

Example 4.1.1. An illustration of the epistemic scenarios from Figure 4.1. (a) A single agent $\mathsf{Ag} = \{a\}$ is considered; and also a single atom $p \in \mathsf{Var}$. Two worlds in W suffice for this example, which are labeled with the formula p or $\neg p$ true at each world. The actual world is represented with an underlined formula, i.e. $\underline{p}$. In the actual world, p holds but the agent wrongly believes that $\neg p$ (the horizontal arrow); moreover, a pictures himself as knowing that $\neg p$ in this world (the reflexive arrow). For example, let $p =$ *the agent is poor*, and let agent a ignore that some change took place, e.g. his bank has gone bankrupt. (b) A similar scenario, with two agents a, b. Agent b, the bank accountant, (knows that he) is the only one to know that p. (c) Rumors about the bank have been circulating, that would imply p. Agent b knows that p and that a

has some opinion about it; whatever this opinion is, a is assumed to take his own opinion as being common knowledge. Actually, a has not heard about the rumors and still believes that $\neg p$. (d) Finally, agent a learns about the rumors; he now ignores whether p, but knows that b knows the answer (top); after a (trusted) truthful public announcement that p, it becomes common knowledge that p (bottom). This announcement, denoted $p!$ in the figure, corresponds to the dynamic operator $[p]$ in PAL-RC

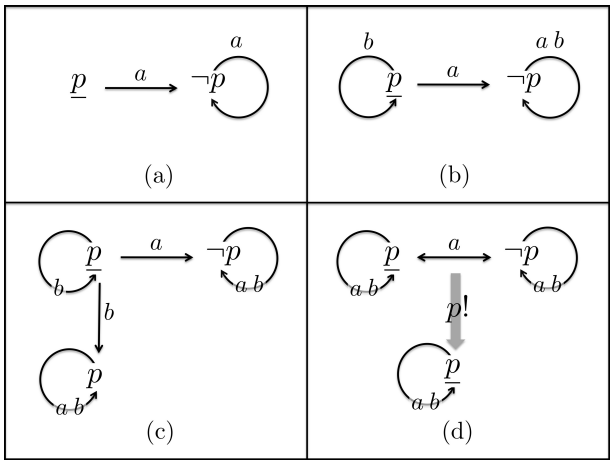


Figure 4.1: An illustration of some scenarios in (dynamic) epistemic logics. (a) single-agent EL; (b)-(c) multi-agent EL; (d) public announcement logic PAL.

Definition 4.1.2 (Epistemic Language; Model; Semantics)**.** The language of epistemic logic EL is a set of formulas defined by

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [a]\varphi$$

An *epistemic model* is a tuple $\langle W, \langle R_a\rangle_{a\in \mathsf{Ag}}, V\rangle$ containing a set of worlds $W \neq \varnothing$, accessibility relations $R_a \in W \times W$ for each agent $a \in \mathsf{Ag}$, and a valuation map $V : \mathsf{Var}W \to \{0, 1\}$. The semantics $M, w \models \varphi$ for a given model M and formula φ are the usual for p, $\wedge$ and $\neg$. For $[a]\varphi$,

$$M, w \models [a]\varphi \quad \text{iff} \quad \text{for all } w' \in W, \quad R_a(w, w') \Rightarrow M, w' \models \varphi$$

The set $\{w \in W \mid M, w \models \varphi\}$ is also denoted $[\![\varphi]\!]^M$.

The expression $\langle a\rangle\varphi$ denotes $\neg[a]\neg\varphi$. The symbols $\rightarrow, \vee, \leftrightarrow, \top, \bot$ are defined as usual from $\{\neg, \wedge\}$. See Figure 4.2 for a list of axioms for the logics of knowledge S5 and belief KD45.

The axiom system for knowledge (S5) corresponds to models where each R_a is an equivalence relation. The logic of belief (KD45) corresponds to models

Taut	propositional tautologies	
(**K**)	$\vdash [a](\varphi \to \psi) \to ([a]\varphi \to [a]\psi)$	*distribution of* $[a]$ *w.r.t.* $\to$
(**T**)	$\vdash [a]\varphi \to \varphi$	*truth*
(**D**)	$\vdash [a]\varphi \to \langle a\rangle\varphi$	*seriality*
(**5**)	$\vdash [a]\varphi \to [a][a]\varphi$	*positive introspection*
(**4**)	$\vdash \neg[a]\varphi \to [a]\neg[a]\varphi$	*negative introspection*
(Nec)	From $\vdash \varphi$ infer $\vdash [a]\varphi$	*necessitation*
(S5)	$=$ Taut $+ \mathbf{K} + \mathbf{T} + \mathbf{5} + \mathbf{4} +$ Nec	*logic of knowledge*
(KD45)	$=$ Taut $+ \mathbf{K} + \mathbf{D} + \mathbf{5} + \mathbf{4} +$ Nec	*logic of belief*

Figure 4.2: (Top) Axioms for epistemic logics. (Bottom) Axiom systems for the logics of knowledge and belief.

where R_a is transitive, serial and Euclidean. Thus, knowledge requires truth **T**, while belief requires consistency **D** (which is equivalent to $\neg[a]\bot$).

Note that epistemic logic describes agents who are ideal reasoners (in this logic), in the sense that they always know what is the logical closure of their beliefs (or, at least, that they can instantly know if a proposition follows from the explicit beliefs).

Group or common notions of knowledge and belief are natural concepts in certain scenarios (conventions, or commonly held beliefs in some community). Group knowledge that φ occurs when each agent in Ag knows the same proposition φ. They can of course wrongly fear that the others do not believe so. Or, in contrast, φ can be publicly assumed, in which case there is common knowledge that φ. These notions also occur in the models of epistemic logic.

Example 4.1.3. In Fig. 4.1 (b), the agents depicted in the non-actual $\neg p$-world have common knowledge that $\neg p$. Since the actual world is p, though, this common knowledge can at most be belief. Moreover, b does not believe it, so it is not even common belief. In summary, there is only a wrong belief by a that $\neg p$ is common knowledge.

The above language of multi-agent epistemic logic EL does not contain modal operators for group or common knowledge (or belief) among all or some subgroup of agents in Ag. In these logics, group knowledge that φ can be expressed with a conjunction $\bigwedge_{a\in\mathsf{Ag}}[a]\varphi$ but common knowledge that φ can only be expressed by an infinite conjunction or set of sentences:

$$\text{common knowledge that } p \quad = \quad \{p, [a]p, [b]p, [a][b]p, [b][a]p, \ldots\}$$

Epistemic logic EL has been extended with modal operators for common knowledge and, more generally, relativized common knowledge (see below).

Dynamic Epistemic Logic

In some scenarios, one might also be interested in modeling information change, rather than just an information state given by some EL-model.

Example 4.1.4. In the top line of Fig. 4.1(d), agent a ignores whether p or $\neg p$. This state of ignorance turns into a state of common knowledge that p after an announcement that p, denoted $p!$ (or $[p]$ in the PAL-RC logic below). The new state is depicted at the bottom of Fig. 4.1(d). The same epistemic state results if agent a publicly observes that p. In either case, b does not learn any new world fact. This agent only updates her knowledge of the epistemic facts.

Since agents are ideal reasoners, information change can only be caused by an external action (or event). In dynamic epistemic logic, the transitions between any two epistemic states caused by a given action are captured with a dynamic modality.

The kind of communicative or sensing actions modeled in dynamic epistemic logics, e.g. PAL, are typically those that correspond to an expansion of the agents' knowledge or beliefs,[1] rather than, say, a revision of this information (like a transition between (b) and (d) from Fig. 4.1). Epistemic actions that cause such expansions in the information of agents are simply represented by world-elimination techniques, as in Fig. 4.1(d).

In the logics of (true) public announcements PAL [117, 65], for example, all agents publicly receive and accept some (true) announcement that φ. An announcement made during a meeting is public if all the agents in Ag attend this meeting. An extension of PAL with a relativized form of common knowledge, called PAL-RC is studied in [139].

Definition 4.1.5 (Language of PAL-RC; Semantics)**.** The language of public announcements with *relativized* common knowledge, PAL-RC consists in

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [a]\varphi \mid [C^{\varphi}]\psi \mid [\varphi]\psi$$

The semantics of EL extends to that of PAL-RC with the new cases:

$$\begin{array}{lll} M, w \models [C^{\varphi}]\psi & \text{iff} & \text{each } [\![\varphi]\!]^M\text{-path from } w \text{ ends in } [\![\psi]\!]^M \\ M, w \models [\varphi]\psi & \text{iff} & M, w \models \varphi \text{ implies } M_{|\varphi}, w \models \psi \end{array}$$

where $M_{|\varphi} = (W', \langle R'_a \rangle_{a \in \mathsf{Ag}}, V')$ is defined by

$$W' = W \cap [\![\varphi]\!]^M, \qquad R'_a = R_a \cap (W' \times W'), \qquad V'(p) = V(p) \cap W$$

Common knowledge that φ is defined as the formula $[C^{\top}]\varphi$.

In Figure 4.1 (d), after this announcement that p, all $\neg p$-worlds in the top model M are eliminated so as to obtain a new model $M_{|\varphi}$ that only contains the p-worlds from M.

[1] From this chapter on, we will use *epistemic* to refer indistinctly to knowledge and belief.

The logic PAL is defined only by the modalities for knowledge $[a]$ and announcements $[\varphi]$. The extensions of the logics EL and PAL with common knowledge $[C^{\top}]$ are called EL-C and PAL-C. See [139] for the axioms of PAL-RC (generalized in those of LCC below), and for a classification of all these logics between EL and PAL-RC in terms of expressivity.

Announcements are *partial* in the sense that they cannot be executed in arbitrary states. For example, truthful announcements that φ can only be made in those worlds where the content φ holds. As a result $\langle\varphi\rangle\top$ is not valid. Moreover, announcements are *functional*, in the sense that $\models \langle\varphi\rangle\psi \rightarrow [\varphi]\psi$. As a consequence, they can be seen as deterministic actions (in terms of epistemic effects).

Logics of announcements PAL, PAL-RC, LCC, etc. ([17, 145, 85, 139]) are in general not closed under uniform substitution, see [77]. That is, some valid formulas for truthful public announcements, like $[p]p$, are not schematically valid. The so-called Moore sentences like $\varphi = p \wedge \neg[a]p$ are examples of this. (This sentence reads, e.g. *p is true but I do not believe it* or *p is true and you do not know it.*) The formula $p \wedge \neg[a]p$ can be true, but it cannot be truthfully announced without changing their truth-value by it. Thus, if in $[p]p$, we replace p by this Moore sentence, we obtain $[p \wedge \neg[a]p]p \wedge \neg[a]p$, which is invalid, so this announcement is unsuccessful. (Moreover, it is $[p\wedge\neg[a]p]\neg(p\wedge\neg[a]p)$ which is valid.) In [77], it is shown that the set of schematic validities in PAL-RC is decidable.

Most proposals in the literature [18], [145], [117] focus on purely epistemic or fact-preserving actions (announcements, sensing). In the literature on planning, in contrast, typical actions are physical and defined by preconditions and post-conditions (or effects). From this point of view, purely epistemic actions can be described as actions with preconditions and trivial post-conditions. For example, p is a in fact precondition for a *truthful (public) announcement that p*, according to the axiom $[p!]\varphi \leftrightarrow (p \rightarrow \varphi)$. Announcements or observations, though, do not change the truth-value of atoms in Var. Different extensions with actions having physical and epistemic effects have been considered, e.g. [142]. The LCC family of Logics of Communication and Change, based on a general notion of action models, has been recently proposed in the literature [139]. The LCC logics are reviewed in Section 4.4. Since these logics are built on top of (an epistemic reading of) propositional dynamic logic PDL, we recall the latter first.

4.2 **Epistemic** PDL

Propositional dynamic logic PDL was proposed to model reasoning about programs (built from basic actions), though, as suggested in [139], PDL programs also admit an epistemic reading if we interpret the basic "program" $[a]$ as the modality for agent a's knowledge; that is, $[a]\varphi$ reads: a knows φ, or a believes φ. The axioms of epistemic PDL do not distinguish between belief and knowledge, as usually understood through the modal logics ($\mathsf{S5}$) and ($\mathsf{KD45}$), respectively. Thus, at the abstract level of PDL we will indistinctly refer to $[a]$ as knowledge

or belief. Within a particular model, though, we can properly refer to one or the other depending on the semantic properties, e.g. whether $[a]\varphi \to \varphi$ holds, etc. Following [139], we refer to PDL under this epistemic reading as E·PDL, in order to avoid confusion.

The syntax of PDL, denoted $\mathcal{L}_{\text{E·PDL}}$, is as follows:

Definition 4.2.1 (E·PDL language)**.** The language of E·PDL, denoted by $\mathcal{L}_{\text{E·PDL}}$, for a given sets of atoms $p \in \mathsf{Var}$ and agents $a \in \mathsf{Ag}$ is the following

$$\begin{array}{lcl} \varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \\ \pi & ::= & a \mid ?\varphi \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

The usual abbreviations apply:

$$\begin{array}{rclcrcl} \bot & = & \neg\top & & \varphi \vee \psi & = & \neg(\neg\varphi \wedge \neg\psi) \\ \langle\pi\rangle\varphi & = & \neg[\pi]\neg\varphi & & \varphi \leftrightarrow \psi & = & (\varphi \to \psi) \wedge (\psi \to \varphi) \end{array}$$

The PDL program constructors: composition ";", choice "$\cup$" and the Kleene star "*" (for the reflexive transitive closure of a relation) allow us to model, among other "epistemic programs",

$[a;b]$	nested beliefs: *agent a believes that b believes that*
$[B]$, or $[a \cup b]$	group belief: *agents in* $B(=\{a,b\})$ *believe that*
$[B^*]$, or $[(a \cup b)^*]$	it is common knowledge among agents in B.

(Thus, the common knowledge operator $[C^\top]$ from Section 4.1 here is denoted $[\mathsf{Ag}^*]$.)

Definition 4.2.2 (Epistemic model)**.** An E·PDL or epistemic model $M = (W, \langle R_a\rangle_{a\in\mathsf{Ag}}, V)$ contains a set of worlds W, a relation R_a in W for each agent a, and an evaluation $V : \mathsf{Var} \to \mathcal{P}(W)$).

While in the usual reading of epistemic logic as S5, the relations R_a are equivalence relations, this restriction is not imposed in the above definition, which uses 'knowledge' in a more general way, including knowledge or belief. The corresponding conditions can be imposed to enforce a particular interpretation, so as to obtain e.g. the original S5 notion of knowledge. The words knowledge and belief are thus used informally and in an interchangeable way.

Definition 4.2.3 (Semantics E·PDL)**.** The semantics of E·PDL for a model $M = (W, \langle R_a\rangle_{a\in\mathsf{Ag}}, V)$, is given by extending the map V into a map $[\![\varphi]\!]^M$ for each formula φ in $\mathcal{L}_{\text{E·PDL}}$:

$$\begin{array}{rclcrcl} [\![\top]\!]^M & = & W & & [\![a]\!]^M & = & R(a) \\ [\![p]\!]^M & = & V(p) & & [\![?\varphi]\!]^M & = & \mathsf{Id}_{[\![\varphi]\!]} \\ [\![\neg\varphi]\!]^M & = & W \setminus [\![\varphi]\!]^M & & [\![\pi_1;\pi_2]\!]^M & = & [\![\pi_1]\!]^M \circ [\![\pi_2]\!]^M \\ [\![\varphi_1\wedge\varphi_2]\!]^M & = & [\![\varphi_1]\!]^M \cap [\![\varphi_2]\!]^M & & [\![\pi_1\cup\pi_2]\!]^M & = & [\![\pi_1]\!]^M \cup [\![\pi_2]\!]^M \\ & & & & [\![\pi^*]\!]^M & = & ([\![\pi]\!]^M)^* \end{array}$$

$$[\![[\pi]\varphi]\!]^M = \{w \in W \mid \forall v((w,v) \in [\![\pi]\!]^M \Rightarrow v \in [\![\varphi]\!]^M\}$$

where $\circ$ is the composition of relations, and $(\llbracket\pi\rrbracket^M)^*$ is the reflexive transitive closure of the binary relation $\llbracket\pi\rrbracket^M$.

Notice in particular that $\llbracket?\bot\rrbracket^M = \emptyset$ and $\llbracket?\top\rrbracket^M = \mathsf{Id}_W$. We recall the axioms and rules that provide a sound and complete axiomatization for E·PDL.

(K)	$[\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$
$(test)$	$[?\varphi_1]\varphi_2 \leftrightarrow (\varphi_1 \to \varphi_2)$
$(sequence)$	$[\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$
$(choice)$	$[\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi$
(mix)	$[\pi^*]\varphi \leftrightarrow \varphi \wedge [\pi][\pi^*]\varphi$, and
$(induction)$	$\varphi \wedge [\pi^*](\varphi \to [\pi]\varphi)) \to [\pi^*]\varphi$.
$(Modus\ ponens)$	From $\vdash \varphi_1$ and $\vdash \varphi_1 \to \varphi_2$, infer $\vdash \varphi_2$,
$(Necessitation)$	From $\vdash \varphi$, infer $\vdash [\pi]\varphi$.

Figure 4.3: Axioms and rules for E·PDL

4.3 Action models U, e

The LCC logic adds to an E·PDL language a set of modalities $[\mathsf{U}, \mathsf{e}]$ for each pointed action model U, e with distinguished (actual) action e. These new operators $[\mathsf{U}, \mathsf{e}]$ read *after the execution of action* e*, it is the case that.*

Definition 4.3.1 (Action model)**.** For a given set of variables Var and agents Ag, an *action model* is a tuple $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$ containing

- $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$, a set of actions
- $\mathsf{R} : \mathsf{Ag} \to (\mathsf{E} \times \mathsf{E})$, a map assigning a relation R_a to each agent $a \in \mathsf{Ag}$
- $\mathsf{pre} : \mathsf{E} \to \mathcal{L}_{\text{E·PDL}}$, a map assigning a precondition $\mathsf{pre}(\mathsf{e})$ to each action e
- $\mathsf{post} : \mathsf{E} \times \mathsf{Var} \to \mathcal{L}_{\text{E·PDL}}$, a map assigning a post-condition $\mathsf{post}(\mathsf{e})(p)$, or $p^{\mathsf{post}(\mathsf{e})}$, to each $\mathsf{e} \in \mathsf{E}$ and $p \in \mathsf{Var}$

Let us fix the above enumeration $\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}$ for the set of actions E, which will be used throughout the next chapters, unless stated otherwise. In particular, this enumeration will fix as well the order of plan search in the next chapter.

Note that the above accessibility relations R_a describe how the execution of an action $\mathsf{e} \in \mathsf{E}$ would appear to a: $\mathsf{e}\mathsf{R}_a\mathsf{f}$ means that if e is executed, agent a will believe it possible that the actual action was f. In that case, $\mathsf{post}(\mathsf{f})(p)$ will contribute to the truth-value of $[\mathsf{U}, \mathsf{e}][a]p$.

The logic of communication and change, or LCC logic, is simply the logic of finite action models U. For the purpose of this thesis, though, we will fix a single action model U, whose logic will be denoted $\mathcal{L}_{\mathsf{U}}$. This model U can be seen as the disjoint union of all the action models needed to model a given scenario.

Definition 4.3.2 (LCC language)**.** The language $\mathcal{L}$of the logic for a given action model U extends that of E·PDL (for the same set of variables Var and agents Ag) with modalities for each pointed action model U, e (i.e. for each $\mathsf{e} \in \mathsf{E}$):

$$\begin{array}{lcl} \varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \mid [\mathsf{U}, \mathsf{e}]\varphi \\ \pi & ::= & a \mid ?\varphi \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

The new modalities $[\mathsf{U}, \mathsf{e}]\varphi$ represent *"after the execution of* e*,* φ *holds"*. The semantics of LCC consists in computing $M, w \models [\mathsf{U}, \mathsf{e}]p$ in terms of the product update of M, w and U, e, defined next.

Definition 4.3.3 (Product Update)**.** Given an E·PDL pointed model M, w and a pointed action model U, e defined for the same sets Var and Ag, we say that $M \circ \mathsf{U}, (w, \mathsf{e})$ is their *product update*, where the model $M \circ \mathsf{U} = (W', \langle R'_a \rangle_{a \in \mathsf{Ag}}, V')$ is defined as follows:

$$\begin{array}{rl} W' = & \{ (w, \mathsf{e}) \in W \times \mathsf{E} \mid M, w \models \mathsf{pre}(\mathsf{e}) \} \\ (w, \mathsf{e}) R'_a (v, \mathsf{f}) \text{ iff} & w R_a v \text{ and } \mathsf{e} \mathsf{R}_a \mathsf{f} \\ V'(p) = & \{ (w, \mathsf{e}) \in W' \mid M, w \models \mathsf{post}(\mathsf{e})(p) \} \end{array}$$

The elements of the updated model $(W', \langle R'_a \rangle_{a \in \mathsf{Ag}}, V')$ will also be denoted as $(W \otimes \mathsf{E}, \langle R_a^{M \circ \mathsf{U}} \rangle_{a \in \mathsf{Ag}}, V^{M \circ \mathsf{U}})$.

Thus, the new states (w, e) capture executions of actions e executable in the previous states w; the truth-value of $\mathsf{post}(\mathsf{e})(p)$ in the latter states w also determines that of p in the new states (w, e).

Definition 4.3.4 (Announcements; Skip action)**.** We define the *skip or null action* skip as follows:

$\mathsf{pre}(\mathsf{skip}) = \top \qquad \mathsf{post}(\mathsf{skip})(p) = p \qquad$ and defined by $\mathsf{skip} \mathsf{R}_b \mathsf{e}$ iff $\mathsf{e} = \mathsf{skip}(\text{any} b \in \mathsf{Ag})$

We say that a *(successful) announcement that* φ made by agent a to some audience B (denoted with a superindex for a and a subindex for B) is:

- *truthful*, denoted $\varphi!^a_B$, iff $\mathsf{pre}(\varphi!^a_B) = \varphi$
- *lying*, denoted $\varphi\dagger^a_B$, iff $\mathsf{pre}(\varphi\dagger^a_B) = \neg\varphi$
- *public* iff $B = \mathsf{Ag}$ and $\varphi!^a_{\mathsf{Ag}} \mathsf{R}_b \mathsf{e} \quad$ iff $\quad \mathsf{e} = \varphi!^a_{\mathsf{Ag}}$, for each $b \in \mathsf{Ag}$ (analogously for $\varphi\dagger^a_{\mathsf{Ag}}$)
- *secret* among B, denoted $\mathsf{U}, \varphi!^a_B$ (similarly for $\varphi\dagger^a_B$), iff

$$\varphi!^a_B \mathsf{R}_b \mathsf{e} \quad \text{iff} \quad \mathsf{e} = \begin{cases} \varphi!^a_B & \text{if } b \in B \cup \{a\} \\ \mathsf{skip} & \text{otherwise} \end{cases}$$

All these announcements f are purely epistemic actions or fact-preserving, in the sense that $\mathsf{post}(\mathsf{f})(p) = p$, for any $p \in \mathsf{Var}$.

These mappings $\mathsf{post}(\mathsf{e})(p)$, also called substitutions and written $p^{\mathsf{post}(\mathsf{e})}$, from $\mathsf{post} : \mathsf{Var} \to \mathcal{L}_{\text{E·PDL}}$ extend to a map $\mathsf{post} : \mathcal{L}_{\text{E·PDL}} \to \mathcal{L}_{\text{E·PDL}}$. A substitution $\mathsf{post}(\mathsf{e}) = \sigma$, that is the map $\sigma : \mathsf{Var} \to \mathcal{L}$ distributes over all symbols. That is, if we define

$$
\begin{array}{llll}
\top^\sigma & = \top & a^\sigma & = a \\
p^\sigma & = \sigma(p) & (?\varphi)^\sigma & = ?\varphi^\sigma \\
(\neg\varphi)^\sigma & = \neg(\varphi^\sigma) & (\pi_1;\pi_2)^\sigma & = \pi_1^\sigma;\pi_2^\sigma \\
(\varphi_1 \wedge \varphi_2)^\sigma & = \varphi_1^\sigma \wedge \varphi_2^\sigma & (\pi_1 \cup \pi_2)^\sigma & = \pi_1^\sigma \cup \pi_2^\sigma \\
([\pi]\varphi)^\sigma & = [\pi^\sigma]\varphi^\sigma & (\pi^*)^\sigma & = (\pi^\sigma)^*
\end{array}
$$

then we have

$$M, w \models \varphi^\sigma \text{ iff } M^\sigma, w \models \varphi \quad \text{and} \quad (w, w') \in [\![\pi^\sigma]\!]^M \text{ iff } (w, w') \in [\![\pi]\!]^{M^\sigma}$$

where the model under substitution $M^\sigma = (W, R, V^\sigma)$ is defined by $V^\sigma(p) = \{w \in W \mid M, w \models \sigma(p)\}$.

The semantics of LCC logics are defined by extending the PDL semantics $[\![\cdot]\!]$ for E·PDL-formulas from Section4.2, with the following case:

$$[\![[\mathsf{U}, \mathsf{e}]\varphi]\!]^M \quad = \quad \{w \in W \mid \text{ if } M, w \models \mathsf{pre}(\mathsf{e}) \text{ then } (w, \mathsf{e}) \in [\![\varphi]\!]^{M \circ \mathsf{U}}\}$$

Remark 4.3.5 (Restricted post-conditions). While in Definition 4.3.1, the post-conditions $\mathsf{post}(\mathsf{e})(p)$ can be assigned any E·PDL-formula, from here on, we will assume that each post-condition $\mathsf{post}(\mathsf{e})(p)$ is restricted to the elements $\{p, \top, \bot\}$.

This restriction was studied in [142] for logics similar to LCC, with epistemic modalities for agents $[a]$ and group common knowledge $[B^*]$ for $B \subseteq \mathsf{Ag}$. The authors show that the logic resulting after this restriction on post-conditions is as expressive as the original where post-conditions are arbitrary formulas.

Later, we will moreover extend this expressivity by introducing a non-deterministic choice operator for actions. Indeed, choice is more general than arbitrary post-conditions φ, since for example it can model random non-determinism, like the toss of a coin. This cannot be done with arbitrary postconditions alone, since we cannot specify necessary and sufficient conditions φ that would result in the coin landing heads. This restriction makes the truth-value of p after e to be either of the following:

$\mathsf{post}(\mathsf{e})(p) = \ldots$	the truth-value of p after e is ...
$\top$	true (since $\top$ is always true, hence true before e)
p	the truth-value of p before the execution of e
$\bot$	false (since $\bot$ is always false)

4.4 Axiom system

In [139], the authors define program transformers $T_{ij}^{\mathsf{U}}(\pi)$ that provide a mapping between E·PDL programs (see Def. 4.4.1). Given any combination of ontic or epistemic actions (e.g. public and private announcements) the transformers provide a complete set of reduction axioms, reducing LCC to E·PDL (see Fig. 6.1).

Definition 4.4.1 (Program transformer)**.** Let an action model U with $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$ be given. The *program transformer* function T_{ij}^{U} on the set of E·PDL programs is defined by:

$$
\begin{aligned}
T^{\mathsf{U}}_{ij}(a) &= \begin{cases} ?\mathsf{pre}(e_i); a & \text{if } \mathsf{e}_i \mathsf{R}(a) \mathsf{e}_j, \\ ?\bot & \text{otherwise} \end{cases} \\
T^{\mathsf{U}}_{ij}(?\varphi) &= \begin{cases} ?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U}, \mathsf{e}_i]\varphi), & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases} \\
T^{\mathsf{U}}_{ij}(\pi_1; \pi_2) &= \textstyle\bigcup_{k=0}^{n-1} (T^{\mathsf{U}}_{ik}(\pi_1); T^{\mathsf{U}}_{kj}(\pi_2)) \\
T^{\mathsf{U}}_{ij}(\pi_1 \cup \pi_2) &= T^{\mathsf{U}}_{ij}(\pi_1) \cup T^{\mathsf{U}}_{ij}(\pi_2) \\
T^{\mathsf{U}}_{ij}(\pi^*) &= K^{\mathsf{U}}_{ijn}(\pi). \\
& \text{where } K^{\mathsf{U}}_{ijn} \text{ is inductively defined as follows:} \\
K^{\mathsf{U}}_{ij0}(\pi) &= \begin{cases} ?\top \cup T^{\mathsf{U}}_{ij}(\pi) & \text{if } i = j \\ T^{\mathsf{U}}_{ij}(\pi) & \text{otherwise} \end{cases} \\
K^{\mathsf{U}}_{ij(k+1)}(\pi) &= \begin{cases} (K^{\mathsf{U}}_{kkk}(\pi))^* & \text{if } i = k = j \\ (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi) & \text{if } i = k \neq j \\ K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^* & \text{if } i \neq k = j \\ K^{\mathsf{U}}_{ijk}(\pi) \cup (K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi)) & \text{if } i \neq k \neq j \end{cases}
\end{aligned}
$$

In a sketch, the new reduction axioms for LCC push the $[\mathsf{U}, \mathsf{e}]$-modalities inside the formula, until the case $[\mathsf{U}, \mathsf{e}]p$ is reached which reduces to the E·PDL formula $\mathsf{pre}(\mathsf{e}) \to \mathsf{post}(\mathsf{e})(p)$.

the axioms and rules for E·PDL	
$[\mathsf{U}, \mathsf{e}]\top \leftrightarrow \top$	(top)
$[\mathsf{U}, \mathsf{e}]p \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \mathsf{post}(e)(p))$	(atoms)
$[\mathsf{U}, \mathsf{e}]\neg\varphi \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U}, \mathsf{e}]\varphi)$	(negation)
$[\mathsf{U}, \mathsf{e}](\varphi_1 \wedge \varphi_2) \leftrightarrow ([\mathsf{U}, \mathsf{e}]\varphi_1 \wedge [\mathsf{U}, \mathsf{e}]\varphi_2)$	(conjunction)
$[\mathsf{U}, \mathsf{e}_i][\pi]\varphi \leftrightarrow \bigwedge_{j=0}^{n-1} [T^{\mathsf{U}}_{ij}(\pi)][\mathsf{U}, \mathsf{e}_j]\varphi$	(E·PDL-programs)
if $\vdash \varphi$ then $\vdash [\mathsf{U}, \mathsf{e}]\varphi$	(Necessitation)

Figure 4.4: A calculus for the LCC logic of finite action models.

Theorem 4.4.2. *[139] The* LCC *logic is sound and complete w.r.t. the axioms of Fig. 6.1.*

The completeness for this calculus is shown by reducing LCC to E·PDL. The translation, simultaneously defined for formulas $t(\cdot)$ and programs $r(\cdot)$ is

$$
\begin{array}{lll}
t(\top) & = \top & r(a) = a \\
t(p) & = p & r(B) = B \\
t(\neg\varphi) & = \neg t(\varphi) & r(?\varphi) = ?t(\varphi) \\
t(\varphi_1 \wedge \varphi_2) & = t(\varphi_1) \wedge t(\varphi_2) & r(\pi_1;\pi_2) = r(\pi_1); r(\pi_2) \\
t([\pi]\varphi) & = [r(\pi)]t(\varphi) & r(\pi_1 \cup \pi_2) = r(\pi_1) \cup r(\pi_2) \\
t([\mathsf{U},\mathsf{e}]\top) & = \top & r(\pi^*) = (r(\pi))^* \\
t([\mathsf{U},\mathsf{e}]p) & = t(\mathsf{pre}(\mathsf{e})) \to p^{\mathsf{post}(\mathsf{e})} & \\
t([\mathsf{U},\mathsf{e}]\neg\varphi) & = t(\mathsf{pre}(\mathsf{e})) \to \neg t([\mathsf{U},\mathsf{e}]\varphi) & \\
t([\mathsf{U},\mathsf{e}](\varphi_1 \wedge \varphi_2)) & = t([\mathsf{U},\mathsf{e}]\varphi) \wedge t([\mathsf{U},\mathsf{e}]\varphi_2) & \\
t([\mathsf{U},\mathsf{e}_i][\pi]\varphi) & = \bigwedge_{j=0}^{n-1} [T_{ij}^{\mathsf{U}}(r(\pi))] t([\mathsf{U},\mathsf{e}_j]\varphi) & \\
t([\mathsf{U},\mathsf{e}][\mathsf{U}',\mathsf{e}']\varphi) & = t([\mathsf{U},\mathsf{e}] t([\mathsf{U}',\mathsf{e}']\varphi)) &
\end{array}
$$

These translation functions t and r will be part of the backward planning algorithms presented in the next sections.

Some basic properties of LCC needed in later results are stated next, e.g. that the actions e are deterministic. Most claims in the next lemma seem to be folklore among the community.

Lemma 4.4.3. *Let* U *be an action model. The following hold in* LCC *for any* $\mathsf{e} \in \mathsf{E}$*:*

$$
\begin{array}{lll}
\textit{(a)} & \models [\mathsf{U},\mathsf{e}] \bigvee_{k\le n} \varphi_k \;\leftrightarrow\; \bigvee_{k\le n} [\mathsf{U},\mathsf{e}]\varphi_k & \textit{for any } \varphi \in \mathcal{L}_{\mathrm{LCC}} \\
\textit{(b)} & \models [\mathsf{U},\mathsf{e}]\varphi \;\leftrightarrow\; (\mathsf{pre}(\mathsf{e}) \to [\mathsf{U},\mathsf{e}]\varphi) & \textit{for any } \varphi \in \mathcal{L}_{\mathrm{E\cdot PDL}} \\
\textit{(c)} & \models [\mathsf{U},\mathsf{e}]\varphi \to (\mathsf{pre}(\mathsf{e}) \to \langle \mathsf{U},\mathsf{e}\rangle\varphi) & \textit{for any } \varphi \in \mathcal{L}_{\mathrm{LCC}} \\
\textit{(c')} & \models \mathsf{pre}(\mathsf{e}) \leftrightarrow \langle \mathsf{U},\mathsf{e}\rangle\top & \\
\textit{(d)} & \models [\mathsf{U},\mathsf{e}]\theta \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \theta^{\mathsf{post}(\mathsf{e})}) & \textit{for any propositional } \theta \\
\textit{(e)} & \models \langle \mathsf{U},\mathsf{e}\rangle\varphi \leftrightarrow \langle \mathsf{U},\mathsf{e}\rangle\top \wedge [\mathsf{U},\mathsf{e}]\varphi & \textit{for any } \varphi \in \mathcal{L}_{\mathrm{LCC}} \\
\textit{(f)} & \models [\mathsf{U},\mathsf{e}](\varphi \to \psi) \;\leftrightarrow\; ([\mathsf{U},\mathsf{e}]\varphi \to [\mathsf{U},\mathsf{e}]\psi) & \textit{(axiom } \mathsf{K}\textit{)}
\end{array}
$$

Proof. For the claim 4.4.3(a), we only show the case $n = 2$. The general case is completely analogous.

$$
\begin{array}{rl}
M, w \models & [\mathsf{U},\mathsf{e}]\varphi_1 \vee \varphi_2 \\
\text{iff } M, w \models & [\mathsf{U},\mathsf{e}]\neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\text{iff } M, w \models & \mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}](\neg\varphi_1 \wedge \neg\varphi_2) \\
\text{iff } M, w \models & \mathsf{pre}(\mathsf{e}) \to \neg([\mathsf{U},\mathsf{e}]\neg\varphi_1 \wedge [\mathsf{U},\mathsf{e}]\neg\varphi_2) \\
\text{iff } M, w \models & \neg\mathsf{pre}(\mathsf{e}) \vee \neg[\mathsf{U},\mathsf{e}]\neg\varphi_1 \vee \neg[\mathsf{U},\mathsf{e}]\neg\varphi_2 \\
\text{iff } M, w \models & \neg\mathsf{pre}(\mathsf{e}) \vee \neg[\mathsf{U},\mathsf{e}]\neg\varphi_1 \vee \neg\mathsf{pre}(\mathsf{e}) \vee \neg[\mathsf{U},\mathsf{e}]\neg\varphi_2 \\
\text{iff } M, w \models & (\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\neg\varphi_1) \vee (\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\neg\varphi_2) \\
\text{iff } M, w \models & [\mathsf{U},\mathsf{e}]\neg\neg\varphi_1 \vee [\mathsf{U},\mathsf{e}]\neg\neg\varphi_2) \\
\text{iff } M, w \models & [\mathsf{U},\mathsf{e}]\varphi_1 \vee [\mathsf{U},\mathsf{e}]\varphi_2
\end{array}
$$

4.4.3(b) This is immediate:

$$
\begin{array}{lll}
& M,w \models \mathsf{pre}(\mathsf{e}) \to [\mathsf{U},\mathsf{e}]\varphi & \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \text{ implies } M,w \models [\mathsf{U},\mathsf{e}]\varphi & \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \text{ implies } w \in [\![[U,\mathsf{e}]\varphi]\!]^{M} & \text{(Def. } [\![\cdot]\!]\text{)} \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \text{ implies} & \\
& \qquad (M,w \models \mathsf{pre}(\mathsf{e}) \text{ implies } (w,\mathsf{e}) \in [\![\varphi]\!]^{M\circ\mathsf{U}}) & \text{(Def. 39 of [139])} \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \text{ implies } (w,\mathsf{e}) \in [\![\varphi]\!]^{M\circ\mathsf{U}} & \\
\text{iff} & w \in [\![[U,\mathsf{e}]\varphi]\!]^{M} & \text{(Def. } [\![\cdot]\!]\text{)} \\
\text{iff} & M,w \models [\mathsf{U},\mathsf{e}]\varphi &
\end{array}
$$

4.4.3(c) Let M,w be arbitrary.

$$
\begin{array}{lllll}
M,w \models & [\mathsf{U},\mathsf{e}]\varphi & & & \\
\text{iff } M,w \models & (\mathsf{pre}(\mathsf{e}) \to \mathsf{pre}(\mathsf{e})) \wedge [\mathsf{U},\mathsf{e}]\varphi & & & \\
\text{iff } M,w \models & (\mathsf{pre}(\mathsf{e}) \to \mathsf{pre}(\mathsf{e})) \wedge (\mathsf{pre}(\mathsf{e}) \to [\mathsf{U},\mathsf{e}]\varphi) & & & \\
\text{iff } M,w \models & \mathsf{pre}(\mathsf{e}) \to (\mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U},\mathsf{e}]\varphi) & & & \\
\text{iff } M,w \not\models & \mathsf{pre}(\mathsf{e}) & \text{or} & M,w \models & \mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U},\mathsf{e}]\varphi \\
\text{iff } M,w \not\models & \mathsf{pre}(\mathsf{e}) & \text{or} & M,w \models & \neg(\neg\mathsf{pre}(\mathsf{e}) \vee \neg[\mathsf{U},\mathsf{e}]\varphi) \\
\text{iff } M,w \not\models & \mathsf{pre}(\mathsf{e}) & \text{or} & M,w \not\models & \neg\mathsf{pre}(\mathsf{e}) \vee \neg[\mathsf{U},\mathsf{e}]\varphi \\
\text{iff } M,w \not\models & \mathsf{pre}(\mathsf{e}) & \text{or} & M,w \not\models & \mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\varphi \\
\text{iff } M,w \not\models & \mathsf{pre}(\mathsf{e}) & \text{or} & M,w \not\models & [\mathsf{U},\mathsf{e}]\neg\varphi \\
\text{iff } M,w \not\models & \mathsf{pre}(\mathsf{e}) & \text{or} & M,w \models & \neg[\mathsf{U},\mathsf{e}]\neg\varphi \\
\text{iff } M,w \models & \mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\neg\varphi & & & \\
\text{iff } M,w \models & \mathsf{pre}(\mathsf{e}) \to \langle\mathsf{U},\mathsf{e}\rangle\varphi & & &
\end{array}
$$

4.4.3(c') For the particular case of $\varphi = \top$, we just add the validity $\models [\mathsf{U},\mathsf{e}]\top$, and thus obtain from (c) that $\models \mathsf{pre}(\mathsf{e}) \to \langle\mathsf{U},\mathsf{e}\rangle\top$. For the other direction, towards a contradiction, let

$$
\begin{array}{lll}
M,w \models & \langle\mathsf{U},\mathsf{e}\rangle\top \wedge \neg\mathsf{pre}(\mathsf{e}) & \\
\text{so } M,w \models & \langle\mathsf{U},\mathsf{e}\rangle\top \wedge (\mathsf{pre}(\mathsf{e}) \to [\mathsf{U},\mathsf{e}]\bot) & \\
\text{iff } M,w \models & \langle\mathsf{U},\mathsf{e}\rangle\top \wedge [\mathsf{U},\mathsf{e}]\bot & \text{(Lemma 4.4.3(a))} \\
\text{iff } M,w \models & \langle\mathsf{U},\mathsf{e}\rangle\top \wedge \neg\langle\mathsf{U},\mathsf{e}\rangle\neg\bot & \\
\text{iff } M,w \models & \langle\mathsf{U},\mathsf{e}\rangle\top \wedge \neg\langle\mathsf{U},\mathsf{e}\rangle\top & \text{(contradiction)}
\end{array}
$$

Thus, for arbitrary M and $w \in W$, we have $M,w \models \mathsf{pre}(\mathsf{e}) \to \langle\mathsf{U},\mathsf{e}\rangle\top$.

4.4.3(d) By induction. We denote by σ the postcondition of e: $\sigma = \mathsf{post}(\mathsf{e})$. (Case p) The reduction axiom for p just gives: $\models [\mathsf{U},\mathsf{e}]p \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to p^{\sigma})$. (Case $\neg\varphi$) Assume (Ind. Hyp.) that $M,w \models [\mathsf{U},\mathsf{e}]\theta$ iff $M,w \models (\mathsf{pre}(\mathsf{e}) \to \theta^{\sigma})$. Then, we have

$$
\begin{array}{lll}
 & M,w \models [\mathsf{U},\mathsf{e}]\neg\theta & \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to \neg([\mathsf{U},\mathsf{e}]\theta) & \text{(LCC axiom for } \neg) \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to \neg(\mathsf{pre}(\mathsf{e}) \to \theta^\sigma) & \text{(Ind. Hyp.)} \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to (\mathsf{pre}(\mathsf{e}) \wedge \neg\theta^\sigma) & \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to \neg\theta^\sigma & \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to (\neg\theta)^\sigma & .
\end{array}
$$

(Case $\theta_1 \wedge \theta_2$) Assume (Ind. Hyp.) that the claim holds for θ_1 and for θ_2. Then

$$
\begin{array}{lll}
 & M,w \models [\mathsf{U},\mathsf{e}]\theta_1 \wedge \theta_2 & \\
\text{iff} & M,w \models [\mathsf{U},\mathsf{e}]\theta_1 \wedge [\mathsf{U},\mathsf{e}]\theta_2 & \\
\text{iff} & M,w \models [\mathsf{U},\mathsf{e}]\theta_1 \text{ and } M,w \models [\mathsf{U},\mathsf{e}]\theta_2 & \text{(LCC axiom for } \wedge) \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to \theta_1^\sigma \text{ and } M,w \models \mathsf{pre}(\mathsf{e}) \to \theta_2^\sigma & \text{(Ind. Hyp.)} \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to (\theta_1^\sigma \wedge \theta_2^\sigma) & \\
\text{iff} & M,w \models \mathsf{pre}(\mathsf{e}) \to (\theta_1 \wedge \theta_2)^\sigma. &
\end{array}
$$

This case concludes the inductive proof for the equivalence between $[\mathsf{U},\mathsf{e}]\theta$ and $\mathsf{pre}(\mathsf{e}) \to \theta^{\mathsf{post}(\mathsf{e})}$.

4.4.3(e) We have the following equivalences:

$$
\begin{array}{lcll}
\models \langle \mathsf{U},\mathsf{e}\rangle\varphi & \leftrightarrow & \neg[\mathsf{U},\mathsf{e}]\neg\varphi & \text{(Def. } \langle \mathsf{U},\mathsf{e}\rangle) \\
\models \langle \mathsf{U},\mathsf{e}\rangle\varphi & \leftrightarrow & \neg(\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\varphi & \text{(Red. axiom } \neg) \\
\models \langle \mathsf{U},\mathsf{e}\rangle\varphi & \leftrightarrow & \mathsf{pre}(\mathsf{e}) \wedge \neg\neg[\mathsf{U},\mathsf{e}]\varphi & \\
\models \langle \mathsf{U},\mathsf{e}\rangle\varphi & \leftrightarrow & \mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U},\mathsf{e}]\varphi & \\
\models \langle \mathsf{U},\mathsf{e}\rangle\varphi & \leftrightarrow & \langle \mathsf{U},\mathsf{e}\rangle\top \wedge [\mathsf{U},\mathsf{e}]\varphi & \text{(Lemma 4.4.3(c'))}
\end{array}
$$

4.4.3(f) Consider the following equivalences

$$
\begin{array}{lll}
 & [\mathsf{U},\mathsf{e}]\varphi \to \psi & \\
\Leftrightarrow & [\mathsf{U},\mathsf{e}]\neg\varphi \vee \psi & \\
\Leftrightarrow & [\mathsf{U},\mathsf{e}]\neg\varphi \vee [\mathsf{U},\mathsf{e}]\psi & \text{Lemma 4.4.3(a)} \\
\Leftrightarrow & (\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\varphi) \vee [\mathsf{U},\mathsf{e}]\psi & \text{Axiom for } \neg \\
\Leftrightarrow & \neg\mathsf{pre}(\mathsf{e}) \vee \neg[\mathsf{U},\mathsf{e}]\varphi \vee [\mathsf{U},\mathsf{e}]\psi & \\
\Leftrightarrow & [\mathsf{U},\mathsf{e}]\varphi \to (\neg\mathsf{pre}(\mathsf{e}) \vee [\mathsf{U},\mathsf{e}]\psi) & \\
\Leftrightarrow & [\mathsf{U},\mathsf{e}]\varphi \to (\mathsf{pre}(\mathsf{e}) \to [\mathsf{U},\mathsf{e}]\psi) & \\
\Leftrightarrow & [\mathsf{U},\mathsf{e}]\varphi \to [\mathsf{U},\mathsf{e}]\psi & \text{Lemma 4.4.3(b)}
\end{array}
$$

□

4.5 Other Approaches

The standard approaches for epistemic logic were started by [76], for the (S5) and (KD45) logics, and [57] for an *interpreted systems* approach, based on temporal rather than dynamic operators. The first contributions in dynamic epistemic logics include [117], [65], [18]. Recent presentations with different kinds of purely epistemic actions can be found in [145], and in the general action

models of [17]. Action models for actions with epistemic and physical effects have also been studied in [144] and [85]. The epistemic modalities considered in these papers (atomic and common knowledge) have been generalized in the LCC framework [139], based instead on an epistemic reading of propositional dynamic logic PDL. As we mentioned, the update semantics in this chapter is based on world-elimination [117, 18]. A different presentation, along the line of [65], considers to defined update semantics for DEL in terms of arrow-elimination instead, see [86, 87]. Several extensions of dynamic epistemic logics DEL are not captured in the LCC family exist as well. See the related work on Chapter 6 for more general classes of action programs.

Also related to the next chapter on planning, quantified versions of the logics of public announcements PAL have been studied in [58]. These logics contain modalities $\diamond\varphi$ expressing the existence of a truthful announcement that would establish the truth of the epistemic formula φ, for example $[\psi]\varphi$. A formula $\diamond\varphi$ can be seen as expressing the existence of a plan solution for an epistemic goal φ.

The assignments in LCC for the description of actions' physical effects $\mathsf{post}(\mathsf{e})$, are defined on the set of atomic propositions Var only. A generalization on this aspect (with assignments to formulas φ) has been considered by the so-called STIT logics [22], which include modalities $[\mathsf{stit}{:}\varphi]\varphi'$ for *after the agent sees to it that φ, it holds that φ'*.

An algebraic approach to dynamic epistemic logics can be found in different contributions [105], [16].

In addition, extensions of DEL or LCC logics with uncertainty have also been studied. These extensions replace the traditional "two-valued" modalities for belief (i.e. belief $[a]$ or disbelief $\neg[a]$) with different kinds of uncertain belief operators. Dynamic epistemic logics with probabilistic epistemic modalities are considered in [84], [138]. Possibilistic extensions of the epistemic modalities are presented in [93].

Some works on combining game-theoretic concepts with dynamic epistemic logics exist in the literature [64], [2]. These mainly deal with the strategic power of some coalition $B \subseteq \mathsf{Ag}$, in the sense of which propositions can they enforce by some suitable strategy. These logics contain modalities $\langle B\rangle\varphi$ expressing that some strategy exists for the group B that forces φ.

Chapter 5

Deterministic Planning in LCC

5.1 Introduction

In this chapter, we introduce planning systems for the fragment of the logic LCC given by an arbitrary action model $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$; the logic will be denoted $\mathcal{L}_{\mathsf{U}}$. The main difference with classical planning is that the planning algorithms

(*initial state*, *available actions*, *goals*)

now can consist of E·PDL *epistemic formulas* (for the goals and the initial state), and *pointed action models* U, e (for the actions).

In this chapter, we drop the usual distinction between a planner and the executing agents. Thus, the (unique) executing agent is the planner agent itself. The actions available to this planner-executioner agent are only a subset $A \subseteq \mathsf{E}$ of the actions in the action model U. Unless stated otherwise, in this chapter we will understand an *action* as an available action in A. Note that non-available actions in $\mathsf{E} \setminus A$ might still play a role for plans, but only by way of available actions.

Example 5.1.1. Assume the planner agent is a magician, and her goal is that *a child believes that the coin vanished into a parallel universe*, as promised. The available action is to *hide the coin fast enough*, but its execution will be seen by the child as an execution of *sending the coin into a parallel universe*, thereby causing the desired belief.

After presenting a planning system for LCC logics, we will study search algorithms for these planning domains. Since the actions $\mathsf{e} \in \mathsf{E}$ in the action model U are deterministic, the planning systems in this chapter are systems for deterministic planning. A deterministic plan is, as usual, an executable sequence of actions in A that necessarily leads from the (any) initial state to some goal state.

Definition 5.1.2 (Planning domain)**.** Given some LCC logic $\mathcal{L}_{\mathsf{U}}$ defined by an action model U, we define a *planning domain* for deterministic planning in $\mathcal{L}_{\mathsf{U}}$ as any triple of the form

$$\mathbb{M} = (\varphi_T, A, \varphi_G)$$

where φ_T, φ_G are consistent E·PDL formulas describing, resp., the initial and goal states; and $A \subseteq \mathsf{E}$ is the subset of a actions available to the agent.

Remark 5.1.3. In this Chapter (and also in Chapter 7), we assume that the planner agent, say a, is one of the agents represented in the language of the corresponding LCC logic; that is, –abusing notation– we assume $a \in \mathsf{Ag}$.

The assumption that the logic modeler –in our case: the planner– represents itself in the object language has been a topic of study in [57] and [11] in the context of (dynamic) epistemic logic. The latter work distinguishes from the external perspective (usually assumed in logical modeling) and the internal perspective of an agent (being modeled in the language). For the sake of simplicity, our planner agent, say a, will assume the usual external perspective. Since in the present case the planner a models itself in her own language $a \in \mathsf{Ag}$, he cannot distinguish between his represented facts and beliefs. In some examples, this can be represented by closing the initial state and/or goals under an $[a]$-modality. (E.g., for any world fact θ or epistemic fact $[b]\theta$, contained among a's goals or initial state, the formulas $[a]\theta$ and $[a][b]\theta$ are also explicitly represented.)

Thus, our planner depicts itself as believing (and only believing) true facts. From the perspective of this planner a, the other agents might entertain false beliefs, as well as ignorance or correct beliefs like he does. Similarly, the planner's goals are represented epistemically $[a]\theta$ or $[a][b]\theta$, rather than as external facts (resp. θ and $[b]\theta$).

5.2 Planning systems for deterministic backward LCC planning

As usual in backward iterative planning, a planner agent considers some open goal and refines its current plan with an action for this goal. As usual, given a goal formula φ (e.g. $\varphi = \varphi_G$), the planner needs to compute the minimal conditions ψ (upon arbitrary states) that would make φ to hold after an execution of e. Thus, after refinement of a plan π with e, this minimal condition ψ will be the new goal replacing φ. More formally, we say $\psi \in \mathcal{L}_{\mathrm{PDL}}$ is the *weakest precondition* for a formula $[\mathsf{U}, \mathsf{e}]\varphi$, iff (in LCC)

$$\models \psi \leftrightarrow [\mathsf{U}, \mathsf{e}]\varphi.$$

In the planning system proposed in this chapter, computing the weakest precondition is done with the help of the translation function t used in the reduction of LCC into E·PDL from Chapter 4.

This notion generalizes the definition of the open goals after some refinement, from e.g. classical planning (see Chapter B). Recall that in classical planning

(without conditional actions), the different variables p, q are logically independent. The logical interactions between propositions in an action model make some such generalization necessary for the purpose of LCC planning.

The weakest precondition for e to cause an arbitrary formula φ is thus identified with the formula:

$$t([\mathsf{U},\mathsf{e}]\varphi \wedge \langle \mathsf{U},\mathsf{e}\rangle \top)$$

Indeed, the correctness of the translation based on t, r makes

$$\models t([\mathsf{U},\mathsf{e}]\varphi \wedge \langle \mathsf{U},\mathsf{e}\rangle \top) \leftrightarrow [\mathsf{U},\mathsf{e}]\varphi \wedge \langle \mathsf{U},\mathsf{e}\rangle \top$$

These functions t, r can then be seen as goal-transforming functions: a current goal φ is mapped into $t([\mathsf{U},\mathsf{e}]\varphi \wedge \langle \mathsf{U},\mathsf{e}\rangle \top)$, which becomes the new goal after we refine the plan with e.

Definition 5.2.1 (Solution)**.** Given some LCC logic for an action model U, and a *planning domain* $\mathbb{M} = (\varphi_T, A, \varphi_G)$, we define a *solution* for $\mathbb{M}$ as a sequence $(\mathsf{f}_1, \ldots, \mathsf{f}_m) \in A^{<\omega}$ of actions in A, such that

$$\models \varphi_T \to [\mathsf{U},\mathsf{f}_1] \ldots [\mathsf{U},\mathsf{f}_m]\varphi_G \qquad \text{and} \qquad \models \varphi_T \to \langle \mathsf{U},\mathsf{f}_1\rangle \ldots \langle \mathsf{U},\mathsf{f}_m\rangle \top$$

The subset $A \subseteq \mathsf{E}$ denotes those actions that are actually available to our planner-executor agent a. Following Example 5.1.1, the reason to distinguish A from E is that some other agent $b \in \mathsf{Ag}$ might attribute our agent a some abilities which a does not actually possess, or that b might fail to attribute a some of her actual abilities. Thus, on the one hand, we want to compute the beliefs of b after an execution of some action e as depending on how b interprets this action e. On the other, we want to make explicit which abilities does our agent possess, in order to build realistic plans.

From here on, π will denote a deterministic plan, i.e. a sequence of actions e in decreasing order of execution (rather than an arbitrary epistemic PDL program as before). Plans are denoted by a pair (*action sequence*, *open goals*).

Definition 5.2.2 (Empty plan; Refinement; Plan; Leaf)**.** Given some planning domain $\mathbb{M} = (\varphi_T, A, \varphi_G)$, the *empty plan* is the pair $\pi_\varnothing = (\varnothing, \varphi_G)$. If $\pi = ((\mathsf{f}', \ldots, \mathsf{f}), \varphi_{\mathsf{goals}(\pi)})$ is a plan, then

$$\pi(\mathsf{e}) = ((\mathsf{f}', \ldots, \mathsf{f}, \mathsf{e}), \varphi_{\mathsf{goals}}(\pi(\mathsf{e})))$$

is also a plan, defined by the goal $\varphi_{\mathsf{goals}(\pi(\mathsf{e}))} = t([\mathsf{U},\mathsf{e}]\varphi_{\mathsf{goals}(\pi)} \wedge \langle \mathsf{U},\mathsf{e}\rangle \top)$. This plan, called the *refinement* of π with e, is also denoted $\pi_\varnothing(\mathsf{f}', \ldots, \mathsf{f}, \mathsf{e})$. Finally, a plan π is a *leaf* iff $\varphi_{\mathsf{goals}(\pi(\mathsf{e}))}$ is inconsistent, or $\models \varphi_{\mathsf{goals}(\pi(\mathsf{e}))} \to \varphi_{\mathsf{goals}(\pi)}$.

Leafs are plans not worth considering, either because (a) when we add the last action refinement e, the resulting plan demands an inconsistent precondition $\varphi_{\mathsf{goals}(\pi(\mathsf{e}))}$ (and hence the plan cannot be executed) or (b) because e does not contribute to delete part of the previous goals $\varphi_{\mathsf{goals}(\pi)}$. The search space for the proposed planning algorithm (see below) is the set sequences $(\mathsf{f}_1, \ldots, \mathsf{f}_m) \in A^{<\omega}$. (These sequences are read in decreasing order of execution, i.e. as the sequence of operators $\mathsf{U}, \mathsf{f}_m, \ldots, \mathsf{U}, \mathsf{f}_1$.) Then, the planning algorithm explores just a fragment of this space, since it will not bother to generate/evaluate further refinements of leaf plans.

5.3 A planning algorithm for deterministic planning in LCC.

A Breadth First Search algorithm for deterministic plans in some LCC logic is defined as follows.

Data: $\mathbb{M} = (\varphi_T, A, \varphi_G)$
Result: π; or fail
initialization: $\pi = \pi_\varnothing$ and $\mathsf{Plans} = \langle \pi \rangle$;
while $\not\models \varphi_T \to \varphi_{\mathsf{goals}(\pi)}$ **do**
  delete π from Plans;
  set $\mathsf{Plans} = \mathsf{Plans}^\frown\langle\, \pi(\mathsf{e}) \mid \mathsf{e} \in A$ and $\pi(\mathsf{e})$ is not a leaf $\rangle$;
  if $\mathsf{Plans} = \varnothing$ **then**
    set $\pi =$ fail
  else
    set $\pi =$ the first element of Plans
  end
end

Algorithm 4: Breadth First Search for backward deterministic planning in LCC.

Recall that the actions $\mathsf{e} \in \mathsf{E}$ -as defined above- are deterministic, in the sense that $\models [\mathsf{U}, \mathsf{e}]\varphi \vee \psi \leftrightarrow ([\mathsf{U}, \mathsf{e}]\varphi \vee [\mathsf{U}, \mathsf{e}]\psi)$. Thus, deterministic plans consists of actions $\mathsf{e} \in A \subseteq \mathsf{E}$ in our current action models U. (Later we will extend LCC with composition $\otimes$ and choice $\cup$ to study the non-deterministic case. There we will fully recover the expressivity of actions defined by arbitrary post-conditions $p^{\mathsf{post}(\mathsf{e})} = \varphi$ of [139], i.e. actions with conditional effects: *if* φ *then (after* e*)* p.)

Theorem 5.3.1. *BFS is sound and complete for* LCC *backward planning: the output* π *of the algorithm in Fig. 4 is a solution for* $(\varphi_T, A, \varphi_G)$*; conversely, if a solution exists, then the algorithm terminates (with a solution output).*

Proof. For Soundness, let us re-enumerate the output $\pi_n = \pi_\varnothing(\mathsf{f}_1, \ldots, \mathsf{f}_m)$ as $\pi_n = \pi_\varnothing(\mathsf{f}_m, \ldots, \mathsf{f}_1)$, so it induces the sequence $[\mathsf{U}, \mathsf{f}_1] \ldots [\mathsf{U}, \mathsf{f}_m]$. We check that the latter is a solution for the input planning domain $\mathbb{M}$. Let us also denote $\pi_k = \pi_\varnothing(\mathsf{f}_m, \ldots, \mathsf{f}_k)$.

We check by induction on the length of the plan that π has these two properties:

$$\text{(S1)}\ \models \varphi_T \to [\mathsf{U}, \mathsf{f}_1] \ldots [\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \qquad \text{(S2)}\ \models \varphi_T \to \langle \mathsf{U}, \mathsf{f}_1 \rangle \ldots \langle \mathsf{U}, \mathsf{f}_k \rangle \top$$

We show (S1)-(S2) by simultaneous induction on the length of the plan.
(Base Case)

(S1) The base case $\models \varphi_T \to [\mathsf{U}_1]\varphi_{\mathsf{goals}(\pi_2)}$ follows from

$$\begin{array}{ll} \models \varphi_T \to \varphi_{\mathsf{goals}(\pi_1)} & \text{(def. of output)} \\ \models \varphi_{\mathsf{goals}(\pi_1)} \to [\mathsf{U}, \mathsf{f}_1]\varphi_{\mathsf{goals}(\pi_2)} & \text{(def. of refinement).} \end{array}$$

These jointly imply our claim.

(S2) The base case $\models \varphi_T \to \langle \mathsf{U}, \mathsf{f}_1 \rangle \top$, reduces to

$$\begin{array}{ll} \text{(i)} & \models \varphi_T \to \varphi_{\mathsf{goals}(\pi_1)}, \text{ and} \\ \text{(ii)} & \models \varphi_{\mathsf{goals}(\pi_1)} \to \langle \mathsf{U}, \mathsf{e} \rangle \top. \end{array}$$

But (i) holds by def. of output for π, and (ii) holds since

$$\varphi_{\mathsf{goals}(\pi_1)} = t([\mathsf{U}, \mathsf{f}_1]\varphi \wedge \langle \mathsf{U}, \mathsf{f}_1 \rangle \top) \text{ implies } \langle \mathsf{U}, \mathsf{f}_1 \rangle \top.$$

(Inductive Case)

(S1) For the claim $\models \varphi_T \to [\mathsf{U}, \mathsf{f}_1][\mathsf{U}, \mathsf{f}_2] \ldots [\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})}$, consider

$$\begin{array}{lll} (1) & \models \varphi_{\mathsf{goals}(\pi_{k+1})} \to [\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})} & \text{(Def. 5.2.1),} \\ (2) & \models [\mathsf{U}, \mathsf{f}_k](\varphi_{\mathsf{goals}(\pi_{k+1})} \to [\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})}) & (1) + \text{Nec.} \\ (3) & \models [\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \to [\mathsf{U}, \mathsf{f}_k][\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})} & (2) + \mathsf{K} \\ (4) & \models [\mathsf{U}, \mathsf{f}_{k-1}]([\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \to [\mathsf{U}, \mathsf{f}_k][\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})}) & (3) + \text{Nec.} \\ \vdots & \vdots & \vdots \end{array}$$

$$\begin{array}{ll} (2k{+}1) & \models [\mathsf{U}, \mathsf{f}_1]([\mathsf{U}, \mathsf{f}_2] \ldots [\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \to [\mathsf{U}, \mathsf{f}_2] \ldots [\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})}) \\ (2k{+}2) & \models [\mathsf{U}, \mathsf{f}_1][\mathsf{U}, \mathsf{f}_2] \ldots [\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \to [\mathsf{U}, \mathsf{f}_1][\mathsf{U}, \mathsf{f}_2] \ldots [\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})} \end{array}$$

Finally, combine the latter with the Ind. Hyp. (S1) for k

$$\models \varphi_T \to [\mathsf{U}, \mathsf{f}_1][\mathsf{U}, \mathsf{f}_2] \ldots [\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})}$$

to obtain the above claim (S1) for $k+1$.

(S2) Consider the previous proof for (S1) but replacing $[\mathsf{U}, \mathsf{f}_{k+1}]\varphi_{\mathsf{goals}(\pi_{k+2})}$ by $\langle \mathsf{U}, \mathsf{f}_{k+1} \rangle \top$. The result is a valid proof for claim (1) below. The proof is completed as follows:

$$\begin{array}{lll} (1) & \models \varphi_T \to [\mathsf{U}, \mathsf{f}_1] \ldots [\mathsf{U}, \mathsf{f}_k]\langle \mathsf{U}, \mathsf{f}_{k+1} \rangle \top & \\ (2) & \models \varphi_T \to \langle \mathsf{U}, \mathsf{f}_1 \rangle \ldots \langle \mathsf{U}, \mathsf{f}_k \rangle \top & \text{(Ind. Hyp. (S2) for } k\text{)} \\ (3) & \models \varphi_T \to \langle \mathsf{U}, \mathsf{f}_1 \rangle \ldots \langle \mathsf{U}, \mathsf{f}_k \rangle \langle \mathsf{U}, \mathsf{f}_{k+1}] \rangle \top & (1), (2) \end{array}$$

The induction proof concludes with the case for m, which is itself a proof that π is a solution, so the algorithm is sound.

For Completeness, let a solution exist for a given planning domain $(\varphi_T, A, \varphi_G)$. Let $[\mathsf{U}, \mathsf{e}_{i_1}], \ldots, [\mathsf{U}, \mathsf{e}_{i_m}]$ be the solution with $\mathsf{e}_{i_1}, \ldots, \mathsf{e}_{i_m}$ in $A^{<\omega}$. Without loss of generality, we can assume this solution: (a) has minimal length and (b) the inverse order $(i_m, \ldots, i_1)$ is lexicographically minimum among other solutions in $A^{<\omega}$ of the same (minimal) length $m-1$. (That is, for each other solution $(\mathsf{e}_{j_0}, \ldots, \mathsf{e}_{j'_m})$ we have $m' > m$, or $m' = m$ and $(i_m, \ldots, i_{k+1}) = (j'_m, \ldots, j_{k+1})$ and $i_k < j_k$, for some $k \leq m$). Let π denote this solution: $\pi = \pi_{\varnothing}(\mathsf{e}_{i_m}, \ldots, \mathsf{e}_{i_1})$. And moreover, redefine each action e_{i_j} as f_j so π becomes $\pi = \pi_{\varnothing}(\mathsf{f}_m, \ldots, \mathsf{f}_1)$.

We proceed to show that π is indeed in the search space and that the BFS algorithm terminates with this solution node π. For this, one must show that

(a) the node π is generated (i.e. each intermediate node $\pi_k = (\mathsf{f}_m, \ldots, \mathsf{f}_k)$ is generated)

(b) for no other node π' with length at most that of π and with $\pi' <_{\text{lex}} \pi$, the plan π' satisfies the Terminating Condition, i.e. $\not\models \varphi_T \to \varphi_{\mathsf{goals}(\pi')}$ for each such π'

Assuming (a), claim (b) is straightforward from the above assumptions on π: assume, towards a contradiction, the contrary of (b). If some other plan π' exists with length at most that of π, with with $\pi' <_{\text{lex}} \pi$, and satisfying the Terminating Condition, then by Soundness π' is a solution, so the above assumption on π fails.

Hence it only remains to show claim (a). This is done by induction. (Base Case) That $\pi_\emptyset$ is generated is obvious by Def. 5.2.2. (Inductive Case) We must show that each refinement $\pi_k = (\mathsf{f}_m, \ldots, \mathsf{f}_{k+1}, \mathsf{f}_k)$ is generated if $\pi_{k+1} = (\mathsf{f}_m, \ldots, \mathsf{f}_{k+1})$ is. To see the inductive case, it suffices to check that the following four claims hold for each $k \leq m$:

(C1) $\models \varphi_{\mathsf{goals}(\pi_k)} \to [\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})}$ (C2) $\models \varphi_{\mathsf{goals}(\pi_k)} \to \langle \mathsf{U}, \mathsf{f}_{k+1}\rangle\top$

(C3) $\varphi_{\mathsf{goals}(\pi_k)}$ is consistent, (C4) $\not\models \varphi_{\mathsf{goals}(\pi_{k+1}(\mathsf{f}_k))} \to \varphi_{\mathsf{goals}(\pi_{k+1})}$

(C1) and (C2) follow from the definition of $\varphi_{\mathsf{goals}(\pi_k)}$ and the correctness of the translation defined by t, r.

For (C3), we need the next auxiliary result:

$$\varphi_{\mathsf{goals}(\pi_k)} \equiv [\mathsf{U}, \mathsf{f}_k] \ldots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \langle \mathsf{U}, \mathsf{f}_k\rangle \ldots \langle \mathsf{U}, \mathsf{f}_m\rangle\top$$

This is shown by induction. (Base Case m) The RHS is simply $[\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \langle \mathsf{U}, \mathsf{f}_m\rangle\top$, which is equivalent to $t([\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \langle \mathsf{U}, \mathsf{f}_m\rangle\top)$. But the latter is simply the LHS $\varphi_{\mathsf{goals}(\pi_m)}$, so we are done. (Ind. Case $k+1 \to k$.) Assume (Ind. Hyp.) that

$$\begin{aligned}
\varphi_{\mathsf{goals}(\pi_{k+1})} &\equiv [\mathsf{U}, \mathsf{f}_{k+1}] \ldots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \ldots \langle \mathsf{U}, \mathsf{f}_m\rangle\top \\
\text{Then,} & \\
\varphi_{\mathsf{goals}(\pi_k)} &= t([\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \wedge \langle \mathsf{U}, \mathsf{f}_k\rangle\top) \\
&\equiv [\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \wedge \langle \mathsf{U}, \mathsf{f}_k\rangle\top) && \text{(correctness of } t\text{)} \\
&\equiv [\mathsf{U}, \mathsf{f}_k]([\mathsf{U}, \mathsf{f}_{k+1}] \ldots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \\
&\qquad \wedge\langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \ldots \langle \mathsf{U}, \mathsf{f}_m\rangle\top) \wedge \\
&\quad \wedge\langle \mathsf{U}, \mathsf{f}_k\rangle\top && \text{(ind. hyp.)} \\
&\equiv [\mathsf{U}, \mathsf{f}_k][\mathsf{U}, \mathsf{f}_{k+1}] \ldots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \\
&\quad \wedge[\mathsf{U}, \mathsf{f}_k]\langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \ldots \langle \mathsf{U}, \mathsf{f}_m\rangle\top \wedge \\
&\quad \wedge\langle \mathsf{U}, \mathsf{f}_k\rangle\top && \text{(Red. Axiom } \wedge\text{)} \\
&\equiv [\mathsf{U}, \mathsf{f}_k][\mathsf{U}, \mathsf{f}_{k+1}] \ldots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \\
&\quad \wedge\langle \mathsf{U}, \mathsf{f}_k\rangle\langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \ldots \langle \mathsf{U}, \mathsf{f}_m\rangle\top && \text{(Lemma 4.4.3(e))}
\end{aligned}$$

Now, rather than showing that $\varphi_{\mathsf{goals}(\pi_k)}$ is consistent, we show by induction that for each $1 \leq k \leq m$ there exists a model, say M, w, such that $M, w \models$

φ_T and also $M, w \models \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle \varphi_{\mathsf{goals}(\pi_k)}$. From this, the claim on the consistency of $\varphi_{\mathsf{goals}(\pi_k)}$ is straightforward.

(Base Case 1) Since φ_T is consistent (by def. of planning domain), let $M, w \models \varphi_T$. Since $\pi = \pi_1$ is a solution, by def. of solution and the previous fact, we obtain that

$$\begin{array}{rl} M, w \models [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_m\rangle \top, & \text{and then} \\ M, w \models \varphi_{\mathsf{goals}(\pi_1)} & \text{(Aux. result above)} \end{array}$$

(Ind. Case $k \to k+1$) Assume (Ind. Hyp.) that $M, w \models \varphi_T$ and that

$$\begin{array}{lll} M, w \models & \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle \varphi_{\mathsf{goals}(\pi_k)} & \text{(Ind. Hyp.)} \\ M, w \models & \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle ([\mathsf{U}, \mathsf{f}_k][\mathsf{U}, \mathsf{f}_{k+1}] \dots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge & \\ & \quad \wedge \langle \mathsf{U}, \mathsf{f}_k\rangle \langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \dots \langle \mathsf{U}, \mathsf{f}_m\rangle \top) & \text{(Aux. result)} \\ M, w \models & \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle ([\mathsf{U}, \mathsf{f}_k][\mathsf{U}, \mathsf{f}_{k+1}] \dots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge & \\ & \quad \wedge [\mathsf{U}, \mathsf{f}_k]\langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \dots \langle \mathsf{U}, \mathsf{f}_m\rangle \top \wedge & \\ & \quad \wedge \langle \mathsf{U}, \mathsf{f}_k\rangle \top) & \text{(Lemma 4.4.3(e))} \\ M, w \models & \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle ([\mathsf{U}, \mathsf{f}_k]([\mathsf{U}, \mathsf{f}_{k+1}] \dots [\mathsf{U}, \mathsf{f}_m]\varphi_G \wedge & \\ & \quad \wedge \langle \mathsf{U}, \mathsf{f}_{k+1} \dots \langle \mathsf{U}, \mathsf{f}_m\rangle \top) \wedge & \\ & \quad \wedge \langle \mathsf{U}, \mathsf{f}_k\rangle \top) & \text{(Red. Axiom } \wedge) \\ M, w \models & \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle ([\mathsf{U}, \mathsf{f}_k]\varphi_{\mathsf{goals}(\pi_{k+1})} \wedge \langle \mathsf{U}, \mathsf{f}_k\rangle \top) & \text{(Aux. result)} \\ M, w \models & \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle \langle \mathsf{U}, \mathsf{f}_k\rangle \varphi_{\mathsf{goals}(\pi_{k+1})} & \text{(Lemma 4.4.3(e))} \end{array}$$

(C4) We finally show the second condition for any refinement in the construction of π not to be a leaf plan. For this, suppose the contrary, i.e. $\models \varphi_{\mathsf{goals}(\pi_{k+1}(\mathsf{f}_k))} \to \varphi_{\mathsf{goals}(\pi_{k+1})}$, for some $0 \leq k$. We show that the solution cannot then be minimal, contradicting the initial assumption.

$$\begin{array}{ll} \models \varphi_T \to [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}]\varphi_{\mathsf{goals}(\pi_{k+1}(\mathsf{f}_k))} & \text{((S1) Soundness)} \\ \models [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}](\varphi_{\mathsf{goals}(\pi_{k+1}(\mathsf{f}_k))} \to \varphi_{\mathsf{goals}(\pi_{k+1})}) & \text{(Nec. on Assumption)} \\ \models \varphi_T \to [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}]\varphi_{\mathsf{goals}(\pi_{k+1})} & (\mathsf{K}) \\ \models [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}](\varphi_{\mathsf{goals}(\pi_{k+1})} \to [\mathsf{U}, \mathsf{f}_{k+1}] \dots [\mathsf{U}, \mathsf{f}_m]\varphi_G) & \text{(Nec. on (C3))} \\ \models \varphi_T \to [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}][\mathsf{U}, \mathsf{f}_{k+1}] \dots [\mathsf{U}, \mathsf{f}_m]\varphi_G) & (\mathsf{K}) \end{array}$$

$$\begin{array}{ll} \models \varphi_T \to [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}]\varphi_{\mathsf{goals}(\pi_{k+1})} & \text{(as above)} \\ \models [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}](\varphi_{\mathsf{goals}(\pi_{k+1})} \to \langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \dots \langle \mathsf{U}, \mathsf{f}_m\rangle \top) & \text{(Nec. on (C3))} \\ \models \varphi_T \to [\mathsf{U}, \mathsf{f}_1] \dots [\mathsf{U}, \mathsf{f}_{k-1}]\langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \dots \langle \mathsf{U}, \mathsf{f}_m\rangle \top & (\mathsf{K}) \\ \models \varphi_T \to \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle \top & \text{((S2) Soundness)} \\ \models \varphi_T \to \langle \mathsf{U}, \mathsf{f}_1\rangle \dots \langle \mathsf{U}, \mathsf{f}_{k-1}\rangle \langle \mathsf{U}, \mathsf{f}_{k+1}\rangle \dots \langle \mathsf{U}, \mathsf{f}_m\rangle \top & \text{(by last two claims)} \end{array}$$

Thus, (C1)-(C4) hold for π and so we are done. □

An example in multi-agent planning in LCC.

An advantage of the multi-agent versions of epistemic and dynamic epistemic logic of having in the use of *multi-agent* dynamic epistemic logics, for both planning problems in cooperative and non-cooperative scenarios. Following the scenarios considered in Chapter 3, some collaborative agents might share some goals, but not the same information. Or, an agent might want to help another agent to reach its own goal.

Example 5.3.2. Suppose agent a wants to help agent b to reach its own goal φ_{G_b}; e.g. agent a temporally adopts φ_{G_b} as its own goal and adds the set A_b of actions available to b to its own set of available actions A_a. Say, moreover, that for economic or pedagogical reasons, this help is to be limited to actions $A'_a \subseteq A_a$ with roughly no cost, e.g. the communication of information. Agent a can search for a plan for φ_{G_b} using both $A'_a \cup A_b$. The actions of agent a in the resulting plan would provide all the necessary information for b to satisfy her own goals.

The use of dynamic epistemic logics as a foundation for planning systems can also shed some light into some linguistic aspects of communication. We present next an example in pragmatics, the area of linguistics related to the intentions of speakers, and the communication of these intentions using speech acts [15, 133], utterances that aim to change the beliefs or goals of agents, and hence their behavior, in a particular direction. For example, descriptive utterances can rather mean, in some contexts, commands (e.g. *the music you play is annoying*) or questions (e.g. *I do not know what are you talking about*), and so on.

Explaining the pragmatic meaning of such communicative actions has been tried with the help of planning tools [31, 45, 46]. In the present case of dynamic epistemic logics, a planner agent can, towards some goal, generate a plan involving some of her communicative actions, and possibly other agents' actions as well. The fact that the goal is not a belief expansion, but e.g. a physical effect, accounts for the fact that the (descriptive) communicative action has not the usual descriptive meaning. Indeed, the planner's goals reveal the actual communicative intention. To illustrate this, consider the following example, where a statement of a true fact can be seen as a demand or an order.

Example 5.3.3 (Coffee and sugar)**.** Suppose our agent a, in the role of a customer, orders agent b some coffee c with sugar s. Agent b expects to get some money m from this service. The conjunction of their individual goals is

$$\varphi_G = [\{a, b\}^*]c \wedge s \wedge m)$$

(or, if the agents trusted each other, $\varphi_G = [a](c \wedge s) \wedge [b]m$).

Agent b absent-mindedly serves the coffee, executing $\mathsf{give}^b_a(c)$, but say he forgets about the sugar by executing instead the available action $\mathsf{skip}^b_a(s)$. All the relevant physical actions are particular instances of the following.

action e	$\mathsf{give}^x_y(p)$	$\mathsf{skip}^x_y(p)$	
$\mathsf{pre}(\mathsf{e})$	$[x]\neg p$	$[x]\neg p$	for $x, y \in \{a, b\}$
$\mathsf{post}(\mathsf{e})(p)$	$\top$	p	for $x, y \in \{a, b\}$
$\mathsf{R}_z(\mathsf{e}, \mathsf{f})$	iff $\mathsf{f} = \mathsf{give}^x_y(p)$	iff $\mathsf{f} = \mathsf{give}^x_y(p)$	for $x, y, z \in \{a, b\}$

In the case of action $\mathsf{give}^a_b(m)$, we can rather let agent a to consider its precondition to be $[a](\neg m \wedge c \wedge s)$, rather than just $[a]\neg m$, if the agent can only pay for some service she received.

At this point of the story, after only the coffee has been served, the initial state becomes

$$\varphi_T = \neg s \wedge [\{a, b\}^*]c \wedge \neg m$$

The planning domain for our planner agent a is $(\varphi_T, A, \varphi_G)$, where

$$A = \{\mathsf{give}^a_b(m), \mathsf{give}^b_a(s)\} \cup \{\varphi!^a_b\}_{\varphi \in \Phi} \qquad \text{for some finite set } \Phi$$

A solution for $\mathtt{M}_a$ is given by the joint plan

$$[\mathsf{U}, (\neg s)!^a_b][\mathsf{U}, \mathsf{give}^b_a(s)][\mathsf{U}, \mathsf{give}^a_b(m)]$$

built according to the following plan steps refinements (i.e. in inverse execution ordering)

plan refinement	open goals
$\varnothing$	$\varphi_G = [\{a, b\}^*]c \wedge s \wedge m)$
$\mathsf{give}^a_b(m)$	$t([\mathsf{U}, \mathsf{give}^a_b(m)]\varphi_G \wedge \langle \mathsf{U}, \mathsf{give}^a_b(m)\rangle \top)$ $\equiv \mathsf{pre}(\mathsf{give}^a_b(m)) \wedge [\{a, b\}^*]c \wedge s$ $\equiv [a]\neg m \wedge [\{a, b\}^*]c \wedge s$
$\mathsf{give}^b_a(s)$	$= \mathsf{pre}(\mathsf{give}^b_a(s)) \wedge [a]\neg m \wedge [\{a, b\}^*]c$ $= [b]\neg s \wedge [a]\neg m \wedge [\{a, b\}^*]c$
$(\neg s)!^a_b$	$\neg s \wedge [a]\neg m \wedge [\{a, b\}^*]c$
term. cond.	since φ_T implies these open goals

In summary, the announcement $\neg s!^a_b$ that there is no sugar, is not made by a with the intention to state a true fact $\neg s$, but in order to get some sugar from b. If agent b is moreover endowed with goal recognition tools(see e.g. [12]), he can grasp that the intention is indeed this one.

5.4 Conclusions and Related Work

In this chapter we studied Breadth First Search as a planning algorithm for problems expressible in some LCC logic. From an abstract point of view, most of the definitions and all the results in this Chapter would not be necessary to show the claim that planning in LCC reduces to logical inference or validity tasks. These tasks can be directly induced by the planning domain $\mathbb{M} = (\varphi_T, A, \varphi_G)$ as follows:

1. select an arbitrary sequence $[\mathsf{U}, \mathsf{e}] \dots [\mathsf{U}, \mathsf{f}]$ with $\mathsf{e}, \dots, \mathsf{f} \in A$
2. check whether $\models \varphi_T \to [\mathsf{U}, \mathsf{e}] \dots [\mathsf{U}, \mathsf{f}]\varphi_G$;

$$\begin{cases} \text{output } [\mathsf{U}, \mathsf{e}] \dots [\mathsf{U}, \mathsf{f}] & \text{if } \textit{yes} \\ \text{repeat 1} & \textit{otherwise} \end{cases}$$

From the point of view of planning, though, this unprincipled search method is not satisfactory, for at least two reasons. First, it does not allow for heuristic search. Second, it is inefficient because this search method needs to compute once and again the same logical (sub-)steps.

In comparison, a backward (or forward) stepwise construction of the solution plan will only compute each basic step once (in terms of weakest preconditions). For example, the step $\pi_\varnothing \mapsto [\mathsf{U}, \mathsf{e}]$, or the step $[\mathsf{U}, \mathsf{e}] \dots [\mathsf{U}, \mathsf{f}] \longmapsto [\mathsf{U}, \mathsf{f}'][\mathsf{U}, \mathsf{e}] \dots [\mathsf{U}, \mathsf{f}]$ is computed once and used in further plan refinements of these plans. This would permit the study of heuristic functions specific to the language of the LCC logics.

In a series of papers by Bolander et al. [28], [13], the authors study Breadth First Search for forward planning in a dynamic epistemic setting. The authors focus on (un)decidability results for planning in different fragments of DEL logics, by distinguishing between the base epistemic logic (for states), and the product update semantics (for state transitions). This permits to see DEL-planning as an epistemic extension of classical logic. Another approach is that of [12], where the different problems of deduction, planning and abduction are studied in a single framework for DEL logics. Finally, see also [141] for an approach based on model checking, and [94] for several semantic results on DEL-planning.

Most of these approaches are based on forward planning. The epistemic character of many actions studied in DEL logics, though, suggests that a backward approach seems better suited for the purpose of planning in these logics. This is mainly due to the weak or non-existing preconditions for many epistemic actions, specially, some communicative and sensing actions can always be performed. Thus, if the set of available (truthful and lying) communicative actions is huge, as in the case of humans, so will be the space of executable actions. Of course, from a backward approach, many communicative actions will cause a given epistemic goal, e.g. $[b]p$ (the actions $p!, (p \wedge q)!, \dots$) but this can be remedied with natural notions of cost (e.g. the length of the announcement). See Chapter 7.5. On the other hand, except for [28], these works are mostly based on DEL logics [145]. Thus, they have less general action models for communicative actions, than those considered in the LCC logics. In [28], a semantic approach to

forward multi-agent planning is built on the action models of [142]. This paper also considers non-deterministic actions, which is the topic of Chapter 7.

A paper dealing with the three related problems of: progression, plan recognition, and regression (or backward epistemic planning) is Aucher [12] for the logics of Baltag and Moss [17]. The issue of progression has also been studied from an algebraic point of view in Baltag et al. [16].

In a different line of research, we find the logics for intentions or motivational attitudes. Inspired by means-ends reasoning [30] (from desires and beliefs to intentions), several logics of intentions have been studied in the literature. Building on the work of Cohen and Levesque [44], Rao and Georgeff [127], [128] developed the BDI formalism for practical reasoning. BDI logics are multi-modal logics with three families of normal modal operators for beliefs B_a, desires or goals D_a and intentions I_a, for each agent $a \in \mathsf{Ag}$. BDI logics extend the branching time logic CTL* [56] with the above modalities. This expressivity would allow in principle for higher-order reasoning, or nested belief-desired modalities.

Different issues have been identified in the alternative semantics (and interaction axioms) studied for these modalities. For example, in [44], if *an agent a believes a proposition it will also have the intention towards this proposition.* In [127], *agent intentions are closed under tautologies and belief implications* (side-effect problem). In contrast, a planning-style treatment of goals (e.g. this chapter) does not take into consideration the logical consequences of these goals, except if they are relevant to the plan. Also, in [103], it is argued that reasoning about intentions has a defeasible or non-monotonic nature, which makes normal modal logics not to be well suited, in principle, to reason about intentions. An extension with defeasible logic [102] (see Section 1.7) is suggested to this end. A STIT-based approach to BDI logics can also be found in [22, 150].

Finally, a work related to BDI, is the KARO formalism, with an emphasis in actions rather than time. This framework also contains (or defines) modalities for possible intentions, abilities, opportunities. In [75], an epistemic dynamic logic EDL is proposed to reason about interactions between action and knowledge, with a focus on planning under partial observability.

While in general these logical languages are more expressive than those of the present approach, an advantage of the latter lies in the interaction semantics for purely epistemic actions, and its use of action models, which naturally capture the (multi-agent) epistemic dimension of epistemic and physical actions.

Chapter 6

LCC with composition and choice

6.1 Introduction

In this chapter we propose an extension of LCC logics $\mathcal{L}_{\mathsf{U}}$ with bounded composition and choice, denoted $\mathrm{LCC}_{\cup\otimes n}$. Both operations map two actions e, f to a new action denoted $\mathsf{e} \otimes \mathsf{f}$ (composition) and $\mathsf{e} \cup \mathsf{f}$ (choice). The resulting actions have the interpretation defined next:

- $\mathsf{e} \otimes \mathsf{f}$ models an execution of e followed by an execution of f, and
- $\mathsf{e} \cup \mathsf{f}$ models non-deterministic actions: each execution of $\mathsf{e} \cup \mathsf{f}$ either instantiates as an execution of e or as an execution of f.

Even if all the actions constructible from an action model U using product and choice are already captured in some action model U' (hence in the original LCC logic), it is not obvious at all which are those actions and action models. In order to make these constructions explicit, we present a detailed account of these constructions from a basic action model. The extension of a logic $\mathcal{L}_{\mathsf{U}}$ will be latter used in the next chapter for the purpose of planning.

Structure of the chapter

In Section 6.2, we first expand any LCC logic with the *composition of exactly n actions*. This includes the product action models U^n, and the corresponding languages , denoted $\mathrm{LCC}_{\otimes n}$, for these product action models. Then, in Section 6.3 we expand the latter models and logics with the composition of *at most n* actions, resp. denoted $\mathsf{U}^{\leq n}$ and $\mathrm{LCC}_{\otimes n}$. Finally, in Section 6.5 we add choice into these logics to obtain the class of logics $\mathrm{LCC}_{\cup\otimes n}$ for non-deterministic actions, used in the next chapter. The semantics for non-deterministic actions $\mathsf{e} \cup \mathsf{f}$ is presented in terms of multi-pointed models, e.g. (w, e) and (w, f), one for each possible realization of the action $\mathsf{e} \cup \mathsf{f}$ upon a state M, w.

A summary of the results for all these logics (in this chapter) is as follows. For the composition of actions, the product action models U^n and $\mathsf{U}^{\leq n}$ are shown equivalent to (the corresponding number of) product updates $\mathsf{U} \circ \cdots \circ \mathsf{U}$ of the original action model U. This permits to reduce any logic $\mathcal{L}_{\mathsf{U}^n}$ or $\mathcal{L}_{\mathsf{U}^{\leq n}}$ to the corresponding LCC base logic $\mathcal{L}_{\mathsf{U}}$ (and hence it permits to reduce these logics to E·PDL). These $\mathrm{LCC}_{\otimes n}$ logics, say $\mathcal{L}_{\mathsf{U}^{\leq n}}$, then, have the same expressivity than the corresponding LCC logics, e.g. $\mathcal{L}_{\mathsf{U}}$. After dealing with product, we introduce the operation of choice $\cup$ into the product action models $\mathsf{U}^{\leq n}$. The resulting class of logics, called $\mathrm{LCC}_{\cup\otimes n}$, reduce again to LCC and E·PDL.

6.2 Update with the product of n actions in U^n.

To define the composition of actions, we simply consider the product of action models $\mathsf{U}_1 \otimes \cdots \otimes \mathsf{U}_k$, for each $k \leq n$, where n denotes the maximum number of compositions allowed in the resulting logic $\mathrm{LCC}_{\otimes n}$. An obvious requirement is that these action models are defined for the same set of variables Var and agents Ag.

We define first action models of the form $\mathsf{U}^n = \mathsf{U}_1 \otimes \cdots \otimes \mathsf{U}_n$ and study them from a semantic point of view. This action model U^n just contains the product of exactly n actions, denoted $\mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_n$, that can be executed one after the other.

Note that, in the next definition, the pre' functions of the product action model U^n are defined in terms of the corresponding functions pre from U, and pre' from $\mathsf{U}^2, \ldots, \mathsf{U}^{n-1}$. From here on, we let $\vec{\mathsf{f}}$ denote some sequence $\mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_k$, also written $\mathsf{f}_1, \ldots, \mathsf{f}_k$, for an appropriate k.

Definition 6.2.1 (Product Action Model)**.** Let $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$ be an action model. We define the *product action model*

$$\mathsf{U}^n \;=\; (\mathsf{E}', \mathsf{R}', \mathsf{pre}', \mathsf{post}')$$

inductively as follows:

$$\begin{array}{rl}
\mathsf{E}' = & \left\{ \mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_n \mid \begin{array}{c} \mathsf{f}_1, \ldots, \mathsf{f}_n \in \mathsf{E} \text{ and} \\ \mathsf{pre}'(\mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_n) \text{ is consistent} \end{array} \right\} \\
\mathsf{R}'_a = & \{\langle (\mathsf{e}, \ldots, \mathsf{e}'), (\mathsf{f}, \ldots, \mathsf{f}') \rangle \mid \mathsf{e} \mathsf{R}_a \mathsf{f} \text{ and } \ldots \text{ and } \mathsf{e}' \mathsf{R}_a \mathsf{f}' \} \\
\mathsf{pre}'(\mathsf{e} \otimes \mathsf{f}) = & \mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U}, \mathsf{e}]\mathsf{pre}(\mathsf{f}) \qquad \text{for the case } n = 2 \\
\mathsf{pre}'(\mathsf{f}_1 \otimes \vec{\mathsf{f}}) = & \mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U}, \mathsf{e}]\mathsf{pre}(\vec{\mathsf{f}}) \\
\mathsf{post}'(\mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_n) = & \begin{cases} \mathsf{post}(\mathsf{f}_k)(p) & \text{if } \mathsf{post}(\mathsf{f}_k)(p) \neq p = \\ & \qquad = \mathsf{post}(\mathsf{f}_{k+1})(p) = \ldots = \mathsf{post}(\mathsf{f}_n)(p) \\ \mathsf{post}(\mathsf{f}_1)(p) & \text{if } \mathsf{post}(\mathsf{f}_1)(p) = \ldots = \mathsf{post}(\mathsf{f}_n)(p) = p \end{cases}
\end{array}$$

The components of the resulting action model U^n are also denoted $\mathsf{U}^n = (\mathsf{E}_n, \mathsf{R}^n, \mathsf{pre}^n, \mathsf{post}^n)$.

Remark 6.2.2. More formally, Definition 6.2.1 should rather make use of the translation function t to define the precondition of a product action

$$\begin{aligned} \mathsf{pre}'(\mathsf{e} \otimes \mathsf{f}) &= \mathsf{pre}(\mathsf{e}) \wedge t([\mathsf{U}, \mathsf{e}]\mathsf{pre}(\mathsf{f})) \\ \mathsf{pre}'(\mathsf{e} \otimes \vec{\mathsf{f}}) &= \mathsf{pre}(\mathsf{e}) \wedge t([\mathsf{U}, \mathsf{e}]\mathsf{pre}'(\vec{\mathsf{f}})) \end{aligned}$$

The reason is that the above definition of pre' in Def. 6.2.1 would not satisfy the condition $\mathsf{pre} : \mathsf{E} \to \mathcal{L}_{\text{E·PDL}}$ for action models in Definition 4.3.1. For the sake of simplicity, we will keep the above notation and use

$$\mathsf{pre}(\mathsf{e}) \wedge t([\mathsf{U}, \mathsf{e}]\mathsf{pre}'(\vec{\mathsf{f}})) \text{ and } \mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U}, \mathsf{e}]\mathsf{pre}'(\vec{\mathsf{f}})$$

interchangeably, when referring to these preconditions.

Note from Def. 6.2.1 that the action $\mathsf{f} \otimes \cdots \otimes \mathsf{f}'$ treats p just as the latest action in this tuple satisfying $\mathsf{post}(\cdot)(p) \neq p$, i.e. as the latest action that might change the truth-value of p. Finally, observe that some tuples, e.g. of the form $\langle \mathsf{e}, \mathsf{f} \rangle \in \mathsf{E}' \in \mathsf{U}^2$ are not defined.

Example 6.2.3. Assume the light switch of the dormitory is next to the door, and the bed is on the other side. Also assume that the room is (always) so untidy that it can only be crossed with the light on. Let p denote this condition: $p =$ *the light is on*; and define the actions $\mathsf{get.to.bed}$ and sleep in an action model U as follows:

e	pre	$\mathsf{post}(\mathsf{e})(p)$	$\mathsf{R}_a(\mathsf{e}, \mathsf{f})$
$\mathsf{get.to.bed}$	p	p	$\mathsf{f} = \mathsf{get.to.bed}$
sleep	$\neg p$	p	$\mathsf{f} = \mathsf{sleep}$

Then the product action model U^2 does not contain the action $\mathsf{get.to.bed} \otimes \mathsf{sleep}$. If we denote this "product action" as $\mathsf{e} \otimes \mathsf{f}$, the reason that it does not exist in U^2 is that its precondition $\mathsf{pre}'(\mathsf{e} \otimes \mathsf{f})$ would have to be

$$p \wedge [\mathsf{U}, \mathsf{e}]\mathsf{pre}(\mathsf{f}) \quad \equiv \quad p \wedge [\mathsf{U}, \mathsf{e}]\neg p \quad \equiv \quad p \wedge (\neg p)^{\mathsf{post}(\mathsf{e})} \quad \equiv \quad p \wedge \neg p \quad \equiv \quad \bot$$

Since this precondition is inconsistent, the action $\mathsf{get.to.bed} \otimes \mathsf{sleep}$ is not in the action model. (In other words, the agent cannot sleep in this dormitory.)

For the sake of simplicity, we will sometimes assume these impossible actions are formally defined, although they of course will never be executable. For the present logical purposes or the latter purpose of planning, it makes no difference that such product actions $\mathsf{e} \otimes \mathsf{f}$ are not defined in U^2, or that they are actions in U^2 defined by impossible preconditions.

Fact 6.2.4. It can be seen by direct inspection that the product action model U^n is indeed an action model, provided U is.

Moreover, we proceed to show that the update of an E·PDL model M by a product action model, say $\mathsf{U} \otimes \mathsf{U}$, reduces to a sequence of updates with the simpler action model, e.g. $(M \circ \mathsf{U}) \circ \mathsf{U}$. With more detail, updating a world w with an action $\mathsf{e} \otimes \mathsf{f}$ is semantically equivalent to updating w with e first, and then updating again with f. We first check this is the case for $\mathsf{U}^2 = \mathsf{U} \otimes \mathsf{U}$.

Lemma 6.2.5. *Let* U *be an action model defined for some set of atoms* Var *and agents* Ag*. Let* M *be an model of a logic* E·PDL *whose language is defined from the same sets* Var *and* Ag*. We have the following isomorphism*

$$M \circ (\mathsf{U} \otimes \mathsf{U}) \quad \cong \quad (M \circ \mathsf{U}) \circ \mathsf{U}.$$

Proof. Let $f : W \times (\mathsf{E} \times \mathsf{E}) \to (W \times \mathsf{E}) \times \mathsf{E}$ be the function defined by

$$f : (w, (\mathsf{e}, \mathsf{e}')) \longmapsto ((w, \mathsf{e}), \mathsf{e}')$$

(States: $W^{M \circ \mathsf{U}^2} \cong W^{(M \circ \mathsf{U}) \circ \mathsf{U}}$)
Let us check that f is a bijection between $W^{M \circ \mathsf{U}^2}$ and $W^{(M \circ \mathsf{U}) \circ \mathsf{U}}$:

$$\begin{array}{ll}
& (w, \mathsf{e} \otimes \mathsf{f}) \in M \circ (\mathsf{E} \otimes \mathsf{E}) \\
\text{iff} & M, w \models \mathsf{pre}'(\mathsf{e} \otimes \mathsf{f}) \\
\text{iff} & M, w \models \mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U}, \mathsf{e}]\mathsf{pre}(\mathsf{f}) \\
\text{iff} & M, w \models \mathsf{pre}(\mathsf{e}) \quad \text{and} \quad (M, w \models \mathsf{pre}(\mathsf{e}) \text{ implies } M \circ \mathsf{U}, (w, \mathsf{e}) \models \mathsf{pre}(\mathsf{f})) \\
\text{iff} & M, w \models \mathsf{pre}(\mathsf{e}) \quad \text{and} \quad M \circ \mathsf{U}, (w, \mathsf{e}) \models \mathsf{pre}(\mathsf{f}) \\
\text{iff} & (w, \mathsf{e}) \in W^{M \circ \mathsf{U}} \quad \text{and} \quad ((w, \mathsf{e}), \mathsf{f}) \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}} \\
\text{iff} & ((w, \mathsf{e}), \mathsf{f}) \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}}
\end{array}$$

(Accessibility relations: $R_a^{M \circ (\mathsf{U} \otimes \mathsf{U})} \cong R_a^{(M \circ \mathsf{U}) \circ \mathsf{U}}$)

$$\begin{array}{lll}
& (w_1, (\mathsf{e}, \mathsf{f})) R_a^{M \circ (\mathsf{U} \otimes \mathsf{U})} (w_2, (\mathsf{e}', \mathsf{f}')) & \\
\text{iff} & w_1 R_a^M w_2 \text{ and } (\mathsf{e}, \mathsf{f}) \mathsf{R}'_a (\mathsf{e}', \mathsf{f}') & (\text{by Def. } R^{M \circ (\mathsf{U} \otimes \mathsf{U})}) \\
\text{iff} & w_1 R_a^M w_2 \text{ and } (\mathsf{e} \mathsf{R}_a \mathsf{e}' \text{ and } \mathsf{f} \mathsf{R}_a \mathsf{f}') & (\text{by Def. } \mathsf{R}') \\
\text{iff} & (w_1 R_a^M w_2 \text{ and } \mathsf{e} \mathsf{R}_a \mathsf{e}') \text{ and } \mathsf{f} \mathsf{R}_a \mathsf{f}' & (\text{re-bracketing}) \\
\text{iff} & (w_1, \mathsf{e}) R_a^{M \circ \mathsf{U}} (w_2, \mathsf{e}') \text{ and } \mathsf{f} \mathsf{R}_a \mathsf{f}' & (\text{by Def. } R^{M \circ \mathsf{U}}) \\
\text{iff} & ((w_1, \mathsf{e}), \mathsf{f}) R_a^{(M \circ \mathsf{U}) \circ \mathsf{U}} ((w_2, \mathsf{e}'), \mathsf{f}') & (\text{by Def. } R^{(M \circ \mathsf{U}) \circ \mathsf{U}'})
\end{array}$$

(Valuations: $V^{M \circ (\mathsf{U} \otimes \mathsf{U})}(p) \cong V^{(M \circ \mathsf{U}) \circ \mathsf{U}}(p)$)

$$\begin{array}{ll}
& ((w, \mathsf{e}), \mathsf{f}) \in V^{M \circ (\mathsf{U} \otimes \mathsf{U})}(p) \\
\text{iff} & ((w, \mathsf{e}), \mathsf{f}) \in (W \times \mathsf{E}) \times \mathsf{E} \text{ and } M, w \models \mathsf{pre}(\mathsf{e}) \text{ and } M \circ \mathsf{U}, (w, \mathsf{e}) \models \mathsf{pre}(\mathsf{f}) \\
& \qquad \text{and } (M \circ \mathsf{U}) \circ \mathsf{U}, ((w, \mathsf{e}), \mathsf{f}) \models p \\
\text{iff} & ((w, \mathsf{e}), \mathsf{f}) \in (W \times \mathsf{E}) \times \mathsf{E} \text{ and } M, w \models \mathsf{pre}(\mathsf{e}) \wedge [\mathsf{U}, \mathsf{e}]\mathsf{pre}(\mathsf{f}) \\
& \qquad \text{and } (M \circ \mathsf{U}), (w, \mathsf{e}) \models p^{\mathsf{post}(\mathsf{f})} \\
\text{iff} & ((w, \mathsf{e}), \mathsf{f}) \in (W \times \mathsf{E}) \times \mathsf{E} \text{ and } M, w \models \mathsf{pre}(\mathsf{e} \otimes \mathsf{f}) \\
& \qquad \text{and } (M \circ \mathsf{U}), (w, \mathsf{e}) \models \begin{cases} p^{\mathsf{post}(\mathsf{f})} & \text{if } \mathsf{post}(\mathsf{f})(p) \neq p \\ p & \text{if } \mathsf{post}(\mathsf{f})(p) = p \end{cases}
\end{array}$$

$$
\begin{array}{ll}
\text{iff} & ((w,\mathsf{e}),\mathsf{f}) \in (W \times \mathsf{E}) \times \mathsf{E} \text{ and } M, w \models \mathsf{pre}(\mathsf{e} \otimes \mathsf{f}) \\
& \qquad\qquad \text{and } \begin{cases} M, w \models p^{\mathsf{post}(\mathsf{f})} & \text{if } \mathsf{post}(\mathsf{f})(p) \neq p \\ (M \circ \mathsf{U}), (w,\mathsf{e}) \models p & \text{if } \mathsf{post}(\mathsf{f})(p) = p \end{cases} \\
\text{iff} & (w,(\mathsf{e} \otimes \mathsf{f})) \in W \times (\mathsf{E} \times \mathsf{E}) \text{ and } M, w \models \mathsf{pre}(\mathsf{e} \otimes \mathsf{f}) \text{ and } M, w \models p^{\mathsf{post}(\mathsf{e} \otimes \mathsf{f})} \\
\text{iff} & (w,(\mathsf{e} \otimes \mathsf{f})) \in W \times (\mathsf{E} \times \mathsf{E}) \text{ and } M, w \models [\mathsf{U}^2, \mathsf{e} \otimes \mathsf{f}]p \\
\text{iff} & (w,(\mathsf{e} \otimes \mathsf{f})) \in V^{M \circ (\mathsf{U} \otimes \mathsf{U})}(p)
\end{array}
$$

□

This isomorphism extends to the valuations of arbitrary formulas and programs.

Corollary 6.2.6. *For each formula φ in the language of $\mathsf{U} \otimes \mathsf{U}$:*

$$(w,(\mathsf{e},\mathsf{f})) \in [\![\varphi]\!]^{M \circ \mathsf{U}^2} \quad \Leftrightarrow \quad ((w,\mathsf{e}),\mathsf{f}) \in [\![\varphi]\!]^{(M \circ \mathsf{U}) \circ \mathsf{U}}$$

Proof. As mentioned above, the language of LCC logic for U^2 does not contain the formula $\mathsf{pre}(\mathsf{e} \otimes \mathsf{f})$, this formula being in the language for the action model U. Still, we can make use of the translation function t from [139] to obtain a formula equivalent to it, in the language of E·PDL (included in the language for U^2).

The proof is by simultaneous induction on programs and formulas. The basic cases were just considered in Lemma 6.2.5, i.e. $[\![p]\!]^{(\cdot)} = V^{(\cdot)}(p)$ and $[\![a]\!]^{(\cdot)} = R_a^{(\cdot)}$. From here on, the proof relies upon the established bijection between $W^{M \circ \mathsf{U}^2}$ and $W^{(M \circ \mathsf{U}) \mathsf{U}}$ also from Lemma 6.2.5.

For programs, the correspondence for $[\![?\varphi]\!]$ is immediate from the Ind. Hyp. on $[\![\varphi]\!]$. The same occurs for $[\![\pi_1;\pi_2]\!]$ and $[\![\pi_1 \cup \pi_2]\!]$ from the Ind. Hyp. on $[\![\pi_1]\!]$ and $[\![\pi_2]\!]$. Finally, the case $[\![\pi^*]\!]$ is also straightforward from the Ind. Hyp. on $[\![\pi]\!]$.

The same holds for formulas, the identity between $M \circ \mathsf{U}^2$ and $(M \circ \mathsf{U}) \circ \mathsf{U}$ of the corresponding sets $[\![\top]\!]^{(\cdot)}$, $[\![\neg\varphi]\!]^{(\cdot)}$ and $[\![\varphi_1 \wedge \varphi_2]\!]^{(\cdot)}$ is clear using the Ind. Hyp., resp., on none, on $[\![\varphi]\!]^{(\cdot)}$ and on $[\![\varphi_1]\!]^{(\cdot)}$ and $[\![\varphi_2]\!]^{(\cdot)}$. The case for $[\![[\pi]\varphi]\!]^{(\cdot)}$, making use of the Ind. Hyp. for $[\![\pi]\!]^{(\cdot)}$ and $[\![\varphi]\!]^{(\cdot)}$, is as follows:

$$
\begin{array}{ll}
& [\![[\pi]\varphi]\!]^{M \circ \mathsf{U}^2} \\
= & \left\{ (w,(\mathsf{e},\mathsf{f})) \in W^{M \circ \mathsf{U}^2} \mid \begin{array}{c} \forall (v,(\mathsf{e}',\mathsf{f}')) \in W^{M \circ \mathsf{U}^2} \\ \text{if } ((w,(\mathsf{e},\mathsf{f})),(v,(\mathsf{e}',\mathsf{f}'))) \in [\![\pi]\!]^{M \circ \mathsf{U}^2} \\ \text{then } (v,(\mathsf{e}',\mathsf{f}')) \in [\![\varphi]\!]^{M \circ \mathsf{U}^2} \end{array} \right\} \\
= & \left\{ ((w,\mathsf{e}),\mathsf{f}) \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}} \mid \begin{array}{c} \forall ((v,\mathsf{e}'),\mathsf{f}') \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}} \\ \text{if } (((w,\mathsf{e}),\mathsf{f}),((v,\mathsf{e}'),\mathsf{f}')) \in [\![\pi]\!]^{(M \circ \mathsf{U}) \circ \mathsf{U}} \\ \text{then } ((v,\mathsf{e}'),\mathsf{f}') \in [\![\varphi]\!]^{(M \circ \mathsf{U}) \circ \mathsf{U}} \end{array} \right\} \\
= & [\![[\pi]\varphi]\!]^{(M \circ \mathsf{U}) \circ \mathsf{U}}
\end{array}
$$

Finally, consider the case of formulas of the form $[\mathsf{U}^2, \mathsf{f}_1 \otimes \mathsf{f}_2]\varphi$.

$$
\begin{aligned}
& [\![[\mathsf{U}, \mathsf{f}_1 \otimes \mathsf{f}_2]\varphi]\!]^{M \circ \mathsf{U}^2} \\
= & \left\{ (w, (\mathsf{e}_1, \mathsf{e}_2)) \in W^{M \circ \mathsf{U}^2} \ : \ \begin{array}{c} M \circ \mathsf{U}^2, (w, (\mathsf{e}_1, \mathsf{e}_2)) \models \mathsf{pre}(\mathsf{f}_1 \otimes \mathsf{f}_2) \ \text{ implies} \\ ((w, (\mathsf{e}_1, \mathsf{e}_2)), (\mathsf{f}_1, \mathsf{f}_2)) \in [\varphi]^{(M \circ \mathsf{U}^2) \circ \mathsf{U}^2} \end{array} \right\} \\
= & \left\{ (w, (\mathsf{e}_1, \mathsf{e}_2)) \in W^{M \circ \mathsf{U}^2} \ : \ \begin{array}{c} M \circ \mathsf{U}^2, (w, (\mathsf{e}_1, \mathsf{e}_2)) \models \mathsf{pre}(\mathsf{f}_1 \otimes \mathsf{f}_2) \ \text{ implies} \\ (((w, (\mathsf{e}_1, \mathsf{e}_2)), \mathsf{f}_1), \mathsf{f}_2) \in [\varphi]^{((M \circ \mathsf{U}^2) \circ \mathsf{U}) \circ \mathsf{U}} \end{array} \right\} \\
= & \left\{ \begin{array}{l} ((w, \mathsf{e}_1), \mathsf{e}_2) \\ \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}} \end{array} : \ \begin{array}{c} (M \circ \mathsf{U}) \circ \mathsf{U}, ((w, \mathsf{e}_1), \mathsf{e}_2) \models \mathsf{pre}(\mathsf{f}_1) \wedge [\mathsf{U}, \mathsf{f}_1]\mathsf{pre}(\mathsf{f}_2) \\ \text{implies } (((w, (\mathsf{e}_1, \mathsf{e}_2)), \mathsf{f}_1), \mathsf{f}_2) \in [\varphi]^{(((M \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}} \end{array} \right\} \\
= & \left\{ \begin{array}{l} ((w, \mathsf{e}_1), \mathsf{e}_2) \\ \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}} \end{array} : \ \begin{array}{c} (M \circ \mathsf{U}) \circ \mathsf{U}, ((w, \mathsf{e}_1), \mathsf{e}_2) \models \mathsf{pre}(\mathsf{f}_1) \text{ and} \\ ((M \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}, (((w, \mathsf{e}_1), \mathsf{e}_2), \mathsf{f}_1) \models \mathsf{pre}(\mathsf{f}_2) \\ \text{imply } (((w, (\mathsf{e}_1, \mathsf{e}_2)), \mathsf{f}_1), \mathsf{f}_2) \in [\varphi]^{(((M \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}} \end{array} \right\} \\
= & \left\{ ((w, \mathsf{e}_1), \mathsf{e}_2) \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}} \ : \ \begin{array}{c} (M \circ \mathsf{U}) \circ \mathsf{U}, ((w, \mathsf{e}_1), \mathsf{e}_2) \models \mathsf{pre}(\mathsf{f}_1) \text{ implies} \\ (\text{ if } ((M \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}, (((w, \mathsf{e}_1), \mathsf{e}_2), \mathsf{f}_1) \models \mathsf{pre}(\mathsf{f}_2) \\ \text{then } (((w, (\mathsf{e}_1, \mathsf{e}_2)), \mathsf{f}_1), \mathsf{f}_2) \in [\varphi]^{(((M \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}} \ ) \end{array} \right\} \\
= & \left\{ ((w, \mathsf{e}_1), \mathsf{e}_2) \in W^{(M \circ \mathsf{U}) \circ \mathsf{U}} \ : \ \begin{array}{c} (M \circ \mathsf{U}) \circ \mathsf{U}, ((w, \mathsf{e}_1), \mathsf{e}_2) \models \mathsf{pre}(\mathsf{f}_1) \text{ implies} \\ ((M \circ \mathsf{U}) \circ \mathsf{U}) \circ \mathsf{U}, (((w, \mathsf{e}_1), \mathsf{e}_2), \mathsf{f}_1) \models [\mathsf{U}, \mathsf{f}_2]\varphi \end{array} \right\} \\
= & \quad [\![[\mathsf{U}, \mathsf{f}_1][\mathsf{U}, \mathsf{f}_2]\varphi]\!]^{(M \circ \mathsf{U}) \circ \mathsf{U}}
\end{aligned}
$$

□

Observe that the proofs of Lemma 6.2.5 and Corollary 6.2.6 do not depend upon the assumption that the two action models are the same. Thus, we can extend this Lemma to the more general case of the product of two action models U, U'.

Corollary 6.2.7. *Let* U, U' *be action models defined on the same sets of variables* Var *and agents* Ag. *Then,* $M \circ (\mathsf{U} \otimes \mathsf{U}') \cong (M \circ \mathsf{U}) \circ \mathsf{U}'$. *Hence,*

$$[\![\varphi]\!]^{M \circ (\mathsf{U} \otimes \mathsf{U}')} \cong [\![\varphi]\!]^{(M \circ \mathsf{U}) \circ \mathsf{U}'} \quad \text{for each } \varphi \text{ in the language of } \mathsf{U} \otimes \mathsf{U}'$$

Before proceeding to the generalization of this Corollary 6.2.7 to any finite number of action models $\mathsf{U}, \ldots, \mathsf{U}'$, we need the claim that the update with an action model U preserves isomorphisms.

Lemma 6.2.8. *If* $M \cong M'$ *are isomorphic epistemic models, and* U *is an action model, then* $M \circ \mathsf{U} \cong M' \circ \mathsf{U}$.

Proof. Let $M = (W, R, V)$ and $M' = (W', R', V')$ be isomorphic models. Let then $f : W \mapsto W'$ be a bijection satisfying $R_a^M(w, v) \Leftrightarrow R_a^{M'}(f(w), f(v))$ and $w \in V^M(p) \Leftrightarrow f(w) \in V^{M'}(p)$. Define the map $f^+ : W \times \mathsf{E} \to W' \times \mathsf{E}$ simply as $f^+(w, \mathsf{e}) = (f(w), \mathsf{e})$. This is clearly a bijection and moreover

$$
\begin{array}{rrcl}
 & R_a^{M\circ\mathsf{U}}((w,\mathsf{e}),(v,\mathsf{f})) & \Leftrightarrow & R_a^M(w,v) \text{ and } R_a^{\mathsf{U}}(\mathsf{e},\mathsf{f}) \\
\Leftrightarrow & R_a^{M'}(f(w),f(v)) \text{ and } R_a^{\mathsf{U}}(\mathsf{e},\mathsf{f}) & \Leftrightarrow & R_a^{M'\circ\mathsf{U}}((f(w),\mathsf{e}),(f(v),\mathsf{f})) \qquad \text{and} \\
\Leftrightarrow & R_a^{M'}(f^+(w,\mathsf{e}),f^+(v,\mathsf{f})) & & \\
 & (w,\mathsf{e}) \in V^{M\circ\mathsf{U}}(p) & \Leftrightarrow & M,w \models \mathsf{pre}(\mathsf{e}) \wedge p^{\mathsf{post}(\mathsf{e})} \\
 \Leftrightarrow & M',f(w) \models \mathsf{pre}(\mathsf{e}) \wedge p^{\mathsf{post}(\mathsf{e})} & \Leftrightarrow & (f(w),\mathsf{e}) \in V^{M'\circ\mathsf{U}}(p) \\
 \Leftrightarrow & f^+(w,\mathsf{e}) \in V^{M'\circ\mathsf{U}}(p) & &
\end{array}
$$

□

The previous Corollary 6.2.7 for the basic case $n = 2$ extends to an arbitrary finite number $n \geq 2$ of actions $\mathsf{f}_1, \ldots, \mathsf{f}_n$. That is, it extends to updates with products of arbitrary n actions taken from a given action model U, or n different action models $\mathsf{U}_1, \ldots, \mathsf{U}_n$.

Corollary 6.2.9. *Let M be an* E·PDL *model for a given set of atoms* Var *and agents* Ag*, and let* $\mathsf{U}_1, \ldots, \mathsf{U}_n$ *be action models defined for the same sets* $\mathsf{Var}, \mathsf{Ag}$*. We have that*

$$M \circ (\mathsf{U}_1 \otimes \cdots \otimes \mathsf{U}_n) \;\cong\; (M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_n$$

Proof. Consider the mapping $(w, \mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_n) \longmapsto ((w, \mathsf{f}_1), \ldots, \mathsf{f}_n)$. This is clearly a bijection between $M \circ \mathsf{U}^n$ and $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}$. The rest of the proof is by induction. (Base Case $n = 2$) This is simply Lemma 6.2.5. (Inductive Case $n \mapsto n+1$) Assume (Ind. Hyp.) the claim $M \circ (\mathsf{U}_1 \otimes \cdots \otimes \mathsf{U}_n) \cong (M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_n$. Then,

$$
\begin{array}{rll}
 & ((M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_n) \circ \mathsf{U}_{n+1} & \\
\cong & (M \circ (\mathsf{U}_1 \otimes \cdots \otimes \mathsf{U}_n)) \circ \mathsf{U}_{n+1} & \text{(Ind. Hyp. and Lemma 6.2.8)} \\
\cong & M \circ (\mathsf{U}_1 \otimes \cdots \otimes \mathsf{U}_n \otimes \mathsf{U}_{n+1}) & \text{(Corollary 6.2.7)}
\end{array}
$$

□

As a particular case, we conclude that $M \circ \mathsf{U}^n \;\cong\; (M \circ \mathsf{U}) \cdots \circ \mathsf{U}$ (n times).

6.3 Update with the product of at most n actions in $\mathsf{U}^{\leq n}$.

Finally, we can define the action model $\mathsf{U}^{\leq n}$ for the product of at most n actions (from a fixed action model U) in terms of the product action models $\mathsf{U}, \mathsf{U}^2, \ldots, \mathsf{U}^n$ previously defined. In order to avoid confusion with the previous notation the accessibility relation R_a for agent a is now written as $R(a)$.

Definition 6.3.1 (Composite Action Model)**.** Let U be an action model and let $\mathsf{U}_1 = \ldots = \mathsf{U}_n (= \mathsf{U})$ be n different copies of U, denoted $\mathsf{U}_k = (\mathsf{E}_k, \mathsf{R}_k, \mathsf{pre}_k, \mathsf{post}_k)$ for each $1 \leq k \leq n$. We define $\mathsf{U}^{\leq n} = (\mathsf{E}^{\leq n}, \mathsf{R}^{\leq n}, \mathsf{pre}^{\leq n}, \mathsf{post}^{\leq n})$ as follows

$$
\begin{array}{rclcrcl}
\mathsf{E}^{\leq n} & = & \bigcup_{k \leq n} \mathsf{E}_k & & \mathsf{pre}^{\leq n} & = & \bigcup_{k \leq n} \mathsf{pre}_k \\
\mathsf{R}^{\leq n}(a) & = & \bigcup_{k \leq n} \mathsf{R}_k(a) & & \mathsf{post}^{\leq n} & = & \bigcup_{k \leq n} \mathsf{post}_k
\end{array}
$$

The *sequence of at most n updates on a model M*, denoted

$$(M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_{\leq n} = (W^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}}, R^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}}, V^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}})$$

can be defined in a straightforward way from each $(M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_k$ product action model.

$$\begin{array}{rcl} W^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}} & = & \bigcup_{k\leq n} W^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_k} \\ R^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}}(a) & = & \bigcup_{k\leq n} R^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_k}(a) \\ V^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}} & = & \bigcup_{k\leq n} V^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_k} \end{array}$$

It can be observed that $\mathsf{U}^{\leq n}$ is an action model; and also that $(M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_{\leq n}$ is an E·PDL model. Moreover, Corollary 6.2.9 for the product of n actions extends to the present case for the product of $\leq n$ actions.

Corollary 6.3.2. *Let* $\mathsf{U}(= \mathsf{U}_1 = \cdots = \mathsf{U}_n)$ *be an action model defined for the sets of atoms* Var *and agents* Ag*, and let* M *be an model for the* E·PDL *logic defined by the same* Var *and* Ag*. Then*

$$M \circ \mathsf{U}^{\leq n} \cong (M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_{\leq n}$$

Proof. Consider the mapping $(w, (\mathsf{f}_1, \ldots, \mathsf{f}_k)) \longmapsto (((w, \mathsf{f}_1), \ldots), \mathsf{f}_k)$ between $W \times E^{\leq n}$ and $((W \times \mathsf{E}) \cdots \times \mathsf{E})$. Note this mapping is the union of mappings between $M \circ \mathsf{U}^k$ and $(M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_k$ from Corollary 6.2.9.

Since the sets $W \times \mathsf{E}^k, W \times \mathsf{E}^{k'}$ are disjoint for each $1 \leq k, k' \leq n$ with $k \neq k'$, Corollary 6.2.9 implies that the current mapping is a bijection between

$$\begin{array}{rcl} W \times \bigcup_{1\leq k\leq n} \mathsf{E}^k & \text{and} & \bigcup_{1\leq k\leq n} W^{((M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_k)} \\ \text{and so between} & & \\ W^{M\circ \mathsf{U}^{\leq n}} & \text{and} & W^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}} \end{array}$$

The rest of the proof, for the relations $R^{\leq n}(a)$ and $R^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}}$, and the valuations $V^{\leq n}(p)$ and $V^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq n}}$ proceeds by induction.
(Base Case $n = 2$) For the accessibility relations, we have that

$$\begin{array}{rcll} & R^{M\circ \mathsf{U}^{\leq 2}}(a) & & \\ = & \{\langle (w, \mathsf{e}), (w', \mathsf{f})\rangle \in (W \times (\mathsf{E} \cup \mathsf{E}^2))^2 \mid (w, w') \in R^M(a) \text{ and } (\mathsf{e}, \mathsf{f}) \in \mathsf{R}^{\leq 2}(a)\} & & \\ = & \{\langle (w, \mathsf{e}), (w', \mathsf{f})\rangle \in (W \times \mathsf{E})^2 \mid (w, w') \in R^M(a) \text{ and } (\mathsf{e}, \mathsf{f}) \in \mathsf{R}(a)\} & & \\ & \cup \left\{ \langle (w, \mathsf{e}_1 \otimes \mathsf{e}_2), (w', \mathsf{f}_1 \otimes \mathsf{f}_2)\rangle \in (W \times \mathsf{E}^2)^2 \mid \begin{array}{c} (w, w') \in R^M(a) \text{ and} \\ \langle (\mathsf{e}_1, \mathsf{e}_2), (\mathsf{f}_1, \mathsf{f}_2)\rangle \in \mathsf{R}_2(a) \end{array} \right\} & & \\ = & R^{M\circ \mathsf{U}}(a) \cup R^{M\circ \mathsf{U}^2}(a) & & \\ \cong & R^{M\circ \mathsf{U}}(a) \cup R^{(M\circ \mathsf{U})\circ \mathsf{U}}(a) & & \text{(by Lemma 6.2.5)} \\ = & R^{(M\circ \mathsf{U}_1)\cdots\circ \mathsf{U}_{\leq 2}} & & \end{array}$$

For the valuations, we also apply Def. 6.3.1 and Lemma 6.2.5 to reason as follows:

$$
\begin{array}{rl}
& V^{M\circ\mathsf{U}^{\leq 2}(p)} \\
= & \{(w,\mathsf{e}) \in W \times (\mathsf{E} \cup \mathsf{E}^2) \mid M,w \models \mathsf{pre}(\mathsf{e}) \wedge p^{\mathsf{post}(\mathsf{e})}\} \\
= & \{(w,\mathsf{e}) \in W \times \mathsf{E} \mid M,w \models \mathsf{pre}(\mathsf{e}) \wedge p^{\mathsf{post}(\mathsf{e})}\} \\
& \cup \{(w,\mathsf{e}) \in W \times \mathsf{E}^2 \mid M,w \models \mathsf{pre}(\mathsf{e}) \wedge p^{\mathsf{post}(\mathsf{e})}\} \\
= & V^{M\circ\mathsf{U}}(p) \cup V^{M\circ\mathsf{U}^2}(p) \\
= & V^{M\circ\mathsf{U}}(p) \cup V^{(M\circ\mathsf{U})\circ\mathsf{U}}(p) \\
= & V^{(M\circ\mathsf{U}_1)\cdots\circ\mathsf{U}_{\leq 2}}(p)
\end{array}
$$

(Ind. Case $n \mapsto n+1$) Assume (Ind. Hyp.) that $M \circ \mathsf{U}^{\leq n} \cong (M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_{\leq n}$. For the relations $R^{M\circ\mathsf{U}^{\leq n}}(a) \cong R^{(M\circ\mathsf{U}_1)\cdots\circ\mathsf{U}_{\leq n}}(a)$, the proof is analogous to that of the Base Case, with the following replacements

$$
\begin{array}{rcl}
\mathsf{U}^2 & \text{by} & \mathsf{U}^{n+1} \\
\mathsf{U}^{\leq 2} & \text{by} & \mathsf{U}^{\leq n+1} \\
(M \circ \mathsf{U}) \circ \mathsf{U} & \text{by} & (M \circ \mathsf{U}_1) \cdots \circ \mathsf{U}_{n+1} \\
M & \text{by} & M \circ \mathsf{U}^{\leq n}
\end{array}
$$

and using the Ind. Hyp. instead of Lemma 6.2.5. For the valuations $V^{M\circ\mathsf{U}^{\leq n}}(p) = V^{(M\circ\mathsf{U}_1)\cdots\circ\mathsf{U}_{\leq n}}(p)$, the proof proceeds similarly by showing that

$$
\begin{array}{rcl}
V^{M\circ\mathsf{U}^{\leq n+1}}(p) & = & V^{M\circ\mathsf{U}^{\leq n}}(p) \cup V^{M\circ\mathsf{U}^{n+1}}(p) \\
\cong \quad V^{M\circ\mathsf{U}^{\leq n}}(p) \cup V^{(M\circ\mathsf{U}_1)\cdots\circ\mathsf{U}_{n+1}}(p) & = & V^{(M\circ\mathsf{U}_1)\cdots\circ\mathsf{U}_{\leq n+1}}(p)
\end{array}
$$

□

6.4 The logic $\mathrm{LCC}_{\otimes n}$ of the action model $\mathsf{U}^{\leq n}$.

A logic of (bounded) product action models $\mathsf{U}^{\leq n}$, also called an $\mathrm{LCC}_{\otimes n}$ logic, will be denoted as the logic $\mathcal{L}_{\mathsf{U}^{\leq n}}$. The language of $\mathcal{L}_{\mathsf{U}^{\leq n}}$ adds a product action modality for each element $\mathsf{f}_1 \otimes \cdots \mathsf{f}_k$ of $\mathsf{E}^{\leq n}$. Without loss of generality, we can assume that these new modal operators are only introduced for those product actions whose preconditions, according to Definition 6.2.1, are consistent E·PDL-formulas.

Definition 6.4.1. Let U be an action model defined for two sets of atoms Var and agents Ag. Let $\mathsf{U}^{\leq n}$ denote the corresponding product action model for some finite n. We define the language $\mathcal{L}_{\mathsf{U}^{\leq n}}$ as follows:

$$
\begin{array}{rcl}
\varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \mid [\mathsf{U}^{\leq n}, \mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_k]\varphi \\
\pi & ::= & a \mid ?\varphi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*
\end{array}
$$

The semantics of updates with a modality $[\mathsf{U}^{\leq n}, \mathsf{e}]$ for some $\mathsf{e} \in \mathsf{E}$ in the original action model U is as in LCC. For the product of $2 \leq k \leq n$ actions, the semantics is a particular case of the product update with the action model U^k. That is,

$$
\begin{array}{lll}
M, w \models [\mathsf{U}^{\leq n}, \mathsf{e}]\varphi & \text{iff} & M, w \models [\mathsf{U}, \mathsf{e}]\varphi \\
M, w \models [\mathsf{U}^{\leq n}, \mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_k]\varphi & \text{iff} & M, w \models \mathsf{pre}(\mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_k) \text{ implies} \\
 & & \quad M \circ \mathsf{U}^k, (w, \mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_k) \models \varphi
\end{array}
$$

The fact that any $\mathrm{LCC}_{\otimes n}$ logic, say $\mathcal{L}_{\mathsf{U}^{\leq n}}$, is also an LCC logic follows from the observation that $\mathsf{U}^{\leq n}$ is an action model. The logic $\mathcal{L}_{\mathsf{U}^{\leq n}}$ is thus an LCC logic defined by a set of modalities of the form

$$E^{\leq n} = \{\mathsf{e}_1, \ldots, \mathsf{e}_m,\ \mathsf{e}_1 \otimes \mathsf{e}_2,\ \ldots,\ \mathsf{e}_1 \otimes \cdots \otimes \mathsf{e}_k,\ \ldots\}$$

The only difference is that the new accessibility relations and valuations are appropriately defined from the basic action model U, rather than being arbitrarily given, as in the action models U.

Thus, a complete axiom system for an $\mathrm{LCC}_{\otimes n}$ logic $\mathcal{L}_{\mathsf{U}^{\leq n}}$ is simply given by the axioms of LCC, now applied to the new modalities $[\mathsf{U}^{\leq n}, \mathsf{e} \otimes \cdots \otimes \mathsf{f}]$. Let us fix an enumeration of the new set of actions (abusing notation, we will use m again) $\mathsf{E}^{\leq n} = \{\mathsf{e}_1, \ldots, \mathsf{e}_m\}$. In the following, we denote arbitrary elements in $\mathsf{E}^{\leq n}$ by the expressions e and e_i.

the axioms and rules for E·PDL	
$[\mathsf{U}^{\leq n}, \mathsf{e}]\top \leftrightarrow \top$	(top)
$[\mathsf{U}^{\leq n}, \mathsf{e}]p \leftrightarrow (\mathsf{pre}(\mathsf{e}) \rightarrow \mathsf{post}(\mathsf{e})(p))$	(atoms)
$[\mathsf{U}^{\leq n}, \mathsf{e}]\neg\varphi \leftrightarrow (\mathsf{pre}(\mathsf{e}) \rightarrow \neg[\mathsf{U}^{\leq n}, \mathsf{e}]\varphi)$	(negation)
$[\mathsf{U}^{\leq n}, \mathsf{e}](\varphi_1 \wedge \varphi_2) \leftrightarrow ([\mathsf{U}^{\leq n}, \mathsf{e}]\varphi_1 \wedge [\mathsf{U}^{\leq n}, \mathsf{e}]\varphi_2)$	(conjunction)
$[\mathsf{U}^{\leq n}, \mathsf{e}_i][\pi]\varphi \leftrightarrow \bigwedge_{j=1}^{m} [T_{ij}^{\mathsf{U}^{\leq n}}(\pi)][\mathsf{U}^{\leq n}, \mathsf{e}_j]\varphi$	(E·PDL-programs)
$[\mathsf{U}^{\leq n}, \mathsf{f}_1 \otimes \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k]\varphi \leftrightarrow [\mathsf{U}^{\leq n}, \mathsf{f}_1][\mathsf{U}^{\leq n}, \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_n]\varphi$	(product)
if $\vdash \varphi$ then $\vdash [\mathsf{U}^{\leq n}, \mathsf{e}]\varphi$	(Necessitation)

Figure 6.1: A calculus for the $\mathrm{LCC}_{\otimes n}$ logic $\mathcal{L}_{\mathsf{U}^{\leq n}}$. This is the logic of a product action model $\mathsf{U}^{\leq n}$ given by an arbitrary action model U.

The proof of the next result, then, is essentially that of [139].

Theorem 6.4.2. *[139] For any action model* U*, the* $\mathrm{LCC}_{\otimes n}$ *logic* $\mathcal{L}_{\mathsf{U}^{\leq n}}$ *is sound and complete w.r.t. the axioms of Fig. 6.1.*

The translation function to reduce $\mathrm{LCC}_{\otimes n}$ formulas to E·PDL formulas is the same than for LCC. We show that the reduction axiom for product $\otimes$, namely,

$$[\mathsf{U}^{\leq n}, \mathsf{f}_1 \otimes \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k]\varphi \ \leftrightarrow\ [\mathsf{U}^{\leq n}, \mathsf{f}_1][\mathsf{U}^{\leq n}, \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_n]\varphi$$

is valid in the $\mathrm{LCC}_{\otimes n}$ logics.

Proposition 6.4.3. *The product axiom is sound:*

$$\models [\mathsf{U}^{\leq n}, \mathsf{f}_1 \otimes \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k]\varphi \leftrightarrow [\mathsf{U}^{\leq n}, \mathsf{f}_1][\mathsf{U}^{\leq n}, \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_n]\varphi$$

Proof. Let M, w be an arbitrary pointed model of LCC. Using Corollary 6.3.2, we obtain the following equivalences:

$M, w \models [\mathsf{U}^{\leq n}, (\mathsf{f}_1 \otimes \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k)]\varphi$

$M, w \models \mathsf{pre}(\mathsf{f}_1 \otimes \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k)$ implies $M \circ \mathsf{U}^{\leq n}, (w, (\mathsf{f}_1 \otimes \cdots \otimes \mathsf{f}_k)) \models \varphi$

$M, w \models \mathsf{pre}(\mathsf{f}_1) \wedge [\mathsf{U}^{\leq n}, \mathsf{f}_1]\mathsf{pre}(\mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k)$
implies $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, ((w, \mathsf{f}_1), \ldots, \mathsf{f}_k) \models \varphi$

$M, w \models \mathsf{pre}(\mathsf{f}_1)$ and $M, w \models [\mathsf{U}^{\leq n}, \mathsf{f}_1]\mathsf{pre}(\mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k)$
imply that $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, ((w, \mathsf{f}_1), \ldots, \mathsf{f}_k) \models \varphi$

$M, w \models \mathsf{pre}(\mathsf{f}_1)$ and
($M, w \models \mathsf{pre}(\mathsf{f}_1)$ implies $M \circ \mathsf{U}^{\leq n}, (w, \mathsf{f}_1) \models \mathsf{pre}(\mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k)$)
imply $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, ((w, \mathsf{f}_1), \ldots, \mathsf{f}_k) \models \varphi$

$M, w \models \mathsf{pre}(\mathsf{f}_1)$ and $M \circ \mathsf{U}^{\leq n}, (w, \mathsf{f}_1) \models \mathsf{pre}(\mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k)$
imply $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, ((w, \mathsf{f}_1), \ldots, \mathsf{f}_k) \models \varphi$

$M, w \models \mathsf{pre}(\mathsf{f}_1)$ and $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, (w, \mathsf{f}_1) \models \mathsf{pre}(\mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k)$
imply $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, ((w, \mathsf{f}_1), \ldots, \mathsf{f}_k) \models \varphi$

$M, w \models \mathsf{pre}(\mathsf{f}_1)$ implies $\begin{pmatrix} \text{if } (M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, (w, \mathsf{f}_1) \models \mathsf{pre}(\mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k) \\ \text{then } (M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, ((w, \mathsf{f}_1), \ldots, \mathsf{f}_k) \models \varphi \end{pmatrix}$

$M, w \models \mathsf{pre}(\mathsf{f}_1)$ implies $(M \circ \mathsf{U}) \cdots \circ \mathsf{U}_{\leq n}, (w, \mathsf{f}_1) \models [\mathsf{U}^{\leq n}, \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k]\varphi$

$M, w \models [\mathsf{U}^{\leq n}, \mathsf{f}_1][\mathsf{U}^{\leq n}, \mathsf{f}_2 \otimes \cdots \otimes \mathsf{f}_k]\varphi$

□

6.5 $\mathrm{LCC}_{\cup \otimes n}$: choice and non-deterministic actions.

In this section we extend the $\mathrm{LCC}_{\otimes n}$ logics of bounded composition with an operator for non-deterministic choice, denoted $\cup$. For a given action model U or $\mathsf{U}^{\leq n}$, this operation maps a pair of actions e, f into a new action $\mathsf{e} \cup \mathsf{f}$. The latter expression denotes an action with indeterminate effects: an execution of $\mathsf{e} \cup \mathsf{f}$ will turn either as an execution of e or as an execution of f.

These actions $\mathsf{e} \cup \mathsf{f}$ read as follows: after an agent (e.g. our planner) decides to execute $\mathsf{e} \cup \mathsf{f}$, it is another external "agent", e.g. the environment or nature in principle, who chooses an actual deterministic action between e and f, thus determining the outcome after a given execution of $\mathsf{e} \cup \mathsf{f}$.

Example 6.5.1. A coin toss action is defined by the choice between the actions of *tossing heads* and *tossing tails*, denoted toss_h and $\mathsf{toss}_{\neg h}$. Some unknown

mechanism in nature randomly selects whether the coin lands heads or tails. Thus, each execution of $\mathsf{toss}_h \cup \mathsf{toss}_{\neg h}$ by the agent becomes an instance of toss_h or an instance of $\mathsf{toss}_{\neg h}$.

The action of unconditionally switching a light button (i.e. no matter its state), is also modeled as a choice between two deterministic actions, switch it on or switch it off. In this case, it is the environment –including the actual state of the light switch– which decides whether we will switch the light on or off.

This kind of non-determinism is called *demonic* in the literature [140], in opposition to the so-called *angelic* non-determinism; in the latter kind it is the executing agent itself who freely selects between an action e or the other f, thus assuming that both actions are individually available and executable. This is not the case for either of the two non-deterministic actions in the example above (button switch, coin toss).

Definition 6.5.2. The language $\mathcal{L}^{\cup}_{\mathsf{U}^{\leq n}}$ of a logic $\mathcal{L}_{\mathsf{U}^{\leq n}}$ with choice is defined (for given sets of atoms Var and agents Ag) as follows,

$$\begin{array}{lcl} \varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \mid [\mathsf{U}^{\leq n}, \mathsf{E}_d]\varphi \\ \pi & ::= & a \mid ?\varphi \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

where E_d is an arbitrary non-empty subset of $\mathsf{E}^{\leq n}$.

For a given set of actions $\mathsf{E}_d = \{\mathsf{e}, \ldots, \mathsf{f}\}$, choice will be indistinctly represented using any of the following notations: E_d, or $\{\mathsf{e}, \ldots, \mathsf{f}\}$ or $\mathsf{e} \cup \ldots \cup \mathsf{f}$.

The presence of different post-conditions (factual change) in LCC actions prevents us to model the choice of actions $\mathsf{e} \cup \mathsf{f}$ as full-fledged actions in the action model, in contrast to the previous case of product actions $\mathsf{e} \otimes \mathsf{f} \in \mathsf{E}^{\leq n}$.

Example 6.5.3 (Cont'd)**.** Consider an action like *tossing a coin* $= \mathsf{toss}_h \cup \mathsf{toss}_{\neg h}$, and the atom $h =$ *the coin lands heads*. It is not possible to define a correct post-condition in $\mathsf{post}(\mathsf{toss}_h \cup \mathsf{toss}_{\neg h})(h)$. This post-condition formula should behave as $\top = \mathsf{post}(\mathsf{toss}_h)$ in some cases, and $\bot = \mathsf{post}(\mathsf{toss}_{\neg h})$ in the other cases. Thus, the desired post-condition for h cannot be expressed as a unique formula. For example,

$$\mathsf{post}\left(\mathsf{toss}_h \cup \mathsf{toss}_{\neg h}\right))(h) \neq \begin{cases} \top & \text{since it would always collapse to } \mathsf{toss}_h \\ \bot & \text{since it would always collapse to } \mathsf{toss}_{\neg h} \\ p & \text{since it would always collapse to } \mathsf{skip} \end{cases}$$

In summary, if a non-deterministic action is in the action model, then post cannot be a map $\mathsf{Var} \longrightarrow \mathcal{L}_{\text{E·PDL}}$. For this reason, we do not let non-deterministic actions $\mathsf{e} \cup \mathsf{f}$ to be actions, but model the semantics of their executions as a multi-pointed semantics (a pointed model for each deterministic component e and f). Let us observe, though, that there are non-deterministic actions which can effectively be modeled as actions in an action model.

Example 6.5.4. In contrast, to the coin toss action from Examples 6.5.1 and 6.5.3, some non-deterministic actions could be considered as elements of the action model. Consider for example, the action of (unconditionally) *switching the light*. Define this as a new action, also denoted $\mathsf{on}\cup\mathsf{off}$, defined in terms of the two deterministic actions; e.g. on is defined by $\mathsf{pre}(\mathsf{on}) = \neg on$ and $\mathsf{post}(\mathsf{on})(on) = \top$; and define off with the opposite precondition on and effect $\mathsf{post}(\mathsf{off})(on) = \bot$. Then, if the post-conditions for an atom p are arbitrary E·PDL formulas (rather than restricted to $\{\top, \bot, p\}$), we can correctly capture this non-deterministic switching action as the following action:

$$\mathsf{pre}(\mathsf{on} \cup \mathsf{off}) = \mathsf{pre}(\mathsf{on}) \vee \mathsf{pre}(\mathsf{off}) \equiv \top \qquad \text{and} \qquad \mathsf{post}(\mathsf{on} \cup \mathsf{off})(on) = \neg on$$

As Example 6.5.3 shows, though, in general non-deterministic actions cannot be modeled as actions in the action model. In consequence when extending LCC (or $\mathrm{LCC}_{\otimes n}$) with choice, there will be no exact coincidence between the actions e in the action model U and the dynamic modalities $[\mathsf{U}, \mathsf{e}]$ in the language of this logic. (The language will contain more actions than the action model.) This contrasts with logics of purely epistemic actions [18, 17], and also with $\mathrm{LCC}_{\otimes n}$ (or LCC) where there is a correspondence (a bijection) between the actions e in the action model U and the dynamic modalities $[\mathsf{U}, \mathsf{e}]$ in the language of the logic $\mathcal{L}_{\mathsf{U}}$.

As suggested in [139], non-deterministic actions are introduced with the help of multi-pointed semantics.

Definition 6.5.5 (Multi-pointed models; Semantics)**.** Given an epistemic model M and an action model U, let $W_d \subseteq W$ and $\mathsf{E}_d = \{\mathsf{f}_1, \ldots, \mathsf{f}_k\} \subseteq \mathsf{E}$. Then M, W_d and U, E_d are multi-pointed models. We define

$$\begin{array}{lll} M, W_d \models \varphi & \text{iff} & M, w \models \varphi \quad \text{for each } w \in W_d \\ M, w \models [\mathsf{U}, \mathsf{E}_d]\varphi & \text{iff} & M \circ \mathsf{U}, \{(w, \mathsf{f}), \ldots, (w, \mathsf{f}')\} \models \varphi \\ & & \text{for each } (w, \mathsf{f}), \ldots, (w, \mathsf{f}') \in W^{M\circ\mathsf{U}} \text{ with } \mathsf{f}, \ldots, \mathsf{f}' \in \mathsf{E}_d \end{array}$$

In other words, this semantics for $[\mathsf{U}, \mathsf{E}_d]$ modalities simply amounts to the semantics of the operators $[\mathsf{U}, \mathsf{f}]$ for each $\mathsf{f} \in \mathsf{E}_d$. That is,

$$M, w \models [\mathsf{U}, \mathsf{E}_d]\varphi \quad \text{iff} \quad \begin{array}{c} M, w \models \mathsf{pre}(\mathsf{f}) \text{ implies } M \circ \mathsf{U}, (w, \mathsf{f}) \models \varphi \\ \text{for each } \mathsf{f} \in \mathsf{E}_d \end{array}$$

Note also that this extends the previous semantics for deterministic actions $\mathsf{e} \in \mathsf{E}$ or $\mathsf{e} \otimes \cdots \otimes \mathsf{f} \in \mathsf{E}^{\leq n}$. These are taken as a particular case of the above semantics with $\mathsf{E}_d = \{\mathsf{e}\}$ or $\mathsf{E}_d = \{\mathsf{e} \otimes \cdots \otimes \mathsf{f}\}$.

The logics extending some LCC logic with bounded product action and choice, will be denoted $\mathcal{L}^{\cup}_{\mathsf{U}^{\leq n}}$, where $\mathsf{U}^{\leq n} = (\mathsf{E}^{\leq n}, \ldots)$ is the product action model from which modalities are defined, i.e. $\mathsf{E}_d \subseteq \mathsf{E}^{\leq n}$. The class of these logics with *choice* and the *product of at most n actions* is denoted $\mathrm{LCC}_{\cup\otimes n}$.

An axiom system a given $\mathrm{LCC}_{\cup\otimes n}$ logic can be given in terms of the previous axioms for $\mathrm{LCC}_{\otimes n}$ and the next reduction axiom for choice, suggested in [139].

the axioms of $\mathrm{LCC}_{\otimes n}$ for the $\mathsf{U}^{\leq n}$-modalities

plus

$[\mathsf{U}^{\leq n}, \mathsf{E}_d]\varphi \leftrightarrow \bigwedge_{\mathsf{e}\in\mathsf{E}_d}[\mathsf{U}^{\leq n}, \mathsf{e}]\varphi$ (*choice*)

Figure 6.2: The axioms and rules for $\mathrm{LCC}_{\cup\otimes n}$.

Proposition 6.5.6. *For any* $\mathrm{LCC}_{\cup\otimes n}$ *logic of some action model* $\mathsf{U}^{\leq n}$*, the reduction axiom for choice is sound. The necessitation rule for the choice modalities* $\vdash \varphi$ implies $\vdash [\mathsf{U}^{\leq n}, \mathsf{E}_d]\varphi$ *is valid.*

Proof. The proof for the necessitation rule is straightforward

$$\begin{array}{lll} \vdash \varphi & \Rightarrow & \vdash [\mathsf{U}^{\leq n}, \mathsf{e}]\varphi, \quad \text{for each } \mathsf{e} \in \mathsf{E}_d\text{; (by the Nec. rule in LCC)} \\ & \Rightarrow & \vdash \bigwedge_{\mathsf{e}\in\mathsf{E}_d}[\mathsf{U}^{\leq n}, \mathsf{e}]\varphi \quad \text{modus ponens with taut.} \end{array}$$

The proof of the validity of the reduction axiom *choice* for $[\mathsf{U}, \mathsf{E}_d]\varphi$ is also straightforward:

$$\begin{array}{ll} & M, w \models [\mathsf{U}^{\leq n}, \mathsf{E}_d]\varphi \\ \text{iff} & \text{for each } (w, \mathsf{e}) \in W^{M\circ\mathsf{U}^{\leq n}}, \quad M \circ \mathsf{U}^{\leq n}, (w, \mathsf{e}) \models \varphi \\ \text{iff} & \text{for each } \mathsf{e} \in \mathsf{E}_d,\ M, w \models \mathsf{pre}(\mathsf{e}) \text{ implies } M \circ \mathsf{U}^{\leq n}, (w, \mathsf{e}) \models \varphi \\ \text{iff} & \text{for each } \mathsf{e} \in \mathsf{E}_d,\ M, w \models [\mathsf{U}^{\leq n}, \mathsf{e}]\varphi \\ \text{iff} & M, w \models \bigwedge_{\mathsf{e}\in\mathsf{E}_d}[\mathsf{U}^{\leq n}, \mathsf{e}]\varphi \end{array}$$

□

A first consequence of Proposition 6.5.6 is the following.

Corollary 6.5.7. *The axioms of* $\mathrm{LCC}_{\otimes n}$ *for a given action model* $\mathsf{U}^{\leq n}$ *are valid when* φ *ranges over the language* $\mathcal{L}^{\cup}_{\mathsf{U}^{\leq n}}$ *(of this action model) with choice.*

The translation function $t : \mathcal{L}_{\mathrm{LCC}} \longrightarrow \mathcal{L}_{\mathrm{E\cdot PDL}}$ can thus be extended into a mapping $t : \mathcal{L}_{\mathrm{LCC}_{\cup\otimes n}} \longrightarrow \mathcal{L}_{\mathrm{E\cdot PDL}}$ if we add the following clause

$$t([\mathsf{U}^{\leq n}, \mathsf{E}_d]\varphi) \;=\; \bigwedge_{\mathsf{e}\in\mathsf{E}_d} t([\mathsf{U}^{\leq n}, \mathsf{e}]\varphi)$$

By Proposition 6.5.6 and Corollary 6.5.7, this extended mapping preserves the correctness of the former translation function from LCC.

Corollary 6.5.8 (Completeness of $\mathrm{LCC}_{\cup\otimes n}$)**.** *For any* $\mathrm{LCC}_{\cup\otimes n}$ *logic* $\mathcal{L}^{\cup}_{\mathsf{U}^{\leq n}}$*, the corresponding axiom system from Fig. 6.2 is sound and complete.*

Proof. Soundness follows from the correctness of the translation function f. Completeness follows from the fact that each $\mathrm{LCC}_{\cup\otimes n}$-formula is equivalent to some E·PDL-formula. □

The fact that no modality $[\mathsf{U}^{\leq n}, \mathsf{E}_d]$ is defined from an action E_d in the action model implies that the preconditions of E_d are not formally defined by the action model. Despite this, the following Lemma shows that a precondition can be naturally associated to each of these E_d modalities.

Lemma 6.5.9. *The following holds:* $\models \langle \mathsf{U}^{\leq n}, \mathsf{E}_d \rangle \top \leftrightarrow \bigvee_{\mathsf{e} \in \mathsf{E}_d} \mathsf{pre}(\mathsf{e})$.

Proof. We show the simple case $\mathsf{E}_d = \{\mathsf{e}, \mathsf{f}\} \subseteq \mathsf{E}^{\leq n}$. The proof for the general case is analogous.

$$\begin{array}{lll} M, w \models & \langle \mathsf{U}^{\leq n}, \mathsf{e} \cup \mathsf{f} \rangle \top \\ M, w \models & \neg [\mathsf{U}^{\leq n}, \mathsf{e} \cup \mathsf{f}] \neg \top \\ M, w \models & \neg ([\mathsf{U}^{\leq n}, \mathsf{e}] \neg \top \wedge [\mathsf{U}^{\leq n}, \mathsf{f}] \neg \top) \\ M, w \models & \neg [\mathsf{U}^{\leq n}, \mathsf{e}] \neg \top \vee \neg [\mathsf{U}^{\leq n}, \mathsf{f}] \neg \top) \\ M, w \models & \langle \mathsf{U}^{\leq n}, \mathsf{e} \rangle \top \vee \langle \mathsf{U}^{\leq n}, \mathsf{f} \rangle \top \\ M, w \models & \mathsf{pre}(\mathsf{e}) \vee \mathsf{pre}(\mathsf{f}) \end{array}$$

□

Thus, the executability of a non-deterministic action $\mathsf{e} \cup \mathsf{f}$ only demands that at least one of its deterministic components e or f is executable. Notice a general notion of *executability of an action* e *in a world* w is given by the condition $M, w \models \langle \mathsf{U}, \mathsf{e} \rangle \top$. Thus, a general notion of precondition is given by the formula $\langle \mathsf{U}, \mathsf{e} \rangle \top$. Note that in the logics LCC, we have indeed that $\mathsf{pre}(\mathsf{e}) \equiv \langle \mathsf{U}, \mathsf{e} \rangle \top$. Thus, as a result of this Lemma, we can extend this general precondition to non-deterministic actions E_d and identify $\mathsf{pre}(\mathsf{E}_d) = \bigvee_{\mathsf{e} \in \mathsf{E}_d} \mathsf{pre}(\mathsf{e})$.

Moreover, if we apply a similar maneuver with the notion of post-condition, we can identify

$$\mathsf{post}(\mathsf{E}_d)(p) = \bigvee_{\mathsf{e} \in \mathsf{E}_d} \mathsf{post}(\mathsf{e})(p)$$

Under these natural readings of $\mathsf{pre}(\mathsf{E}_d)$ and $\mathsf{post}(\mathsf{E}_d)(p)$, the axioms of LCC somehow "extend" to the present logics. For example, observe that one of the directions of the LCC reduction axiom for *atoms* is preserved

Fact 6.5.10. Under these definitions of $\mathsf{pre}(\mathsf{E}_d)$ and $\mathsf{post}(\mathsf{E}_d)$,

$$[\mathsf{U}^{\leq n}, \mathsf{E}_d] p \quad \rightarrow \quad \mathsf{pre}(\mathsf{E}_d) \rightarrow p^{\mathsf{post}(\mathsf{E}_d)}$$

If, moreover, for each $\mathsf{e}, \mathsf{f} \in \mathsf{E}_d$ the preconditions are the same $\mathsf{pre}(\mathsf{e}) = \mathsf{pre}(\mathsf{f})$, then axiom LCC becomes valid under this reading:

$$[\mathsf{U}^{\leq n}, \mathsf{E}_d] p \quad \leftrightarrow \quad \mathsf{pre}(\mathsf{E}_d) \rightarrow p^{\mathsf{post}(\mathsf{E}_d)}$$

6.6 Conclusions and Related Work

In this chapter, we extended an arbitrary LCC logic of an action model U with the so-called program constructors of composition $\otimes$ and non-deterministic choice $\cup$. A study of non-deterministic actions with composition and choice can be found in [142]. The pointed action models are defined on top of epistemic logic with common knowledge EL-C. The multi-pointed semantics for non-deterministic actions was suggested in [131] and [139]. The present definitions of composition and choice are taken from these papers [142, 131, 139].

Dynamic epistemic logics with the Kleene star or iteration operator * program constructor have also been studied in the literature. An important result in [98] shows the undecidability of the logics of public announcements PAL with program iteration. In light of these results, it seems that the present class of $\mathrm{LCC}_{\cup\otimes n}$ logics are, roughly, the most expressive ones in terms of the usual program constructors. (See [9] for another approach, and the discussion in 7.5.)

Still, there is room for extensions of LCC. In addition to these existing extensions of DEL logics listed in Section 4.5, we might consider extending LCC with further program constructors. Some of these constructors have already been considered in the literature. For example, extensions of DEL logics with concurrent actions have been studied in [143] and [53]. These extensions permit to model scenarios where one or different agents can execute actions in parallel, as we did in Part I of this thesis. The planning systems presented in Chapters 5 and 7 assume, on the contrary, that actions are executed one at a time. This is a fair assumption for many epistemic actions, like communicative actions, but less realistic for physical actions.

On the other hand, different works on the literature have also considered non-determinism in dynamic epistemic logics. See for example [51] for a study of demonic non-determinism in PDL. See also [140] for a study on these topics in a dynamic logic for agents with abilities, and [132] for a similar study in the logic ADL. The present treatment of choice for basic non-deterministic actions is along the line suggested in [139] for LCC.

Chapter 7

Non-Deterministic Planning in LCC

7.1 Introduction

In this chapter, we propose and study a non-deterministic planning system based on an arbitrary logic in the $\mathrm{LCC}_{\cup\otimes n}$ family. The set A of actions in the planning domain $\mathbb{M}$ will now contain actions E_d with disjunctive effects, e.g. $p \vee q$, where neither disjunct p nor q is a necessary effect. More formally,

$$\models [\mathsf{U}^{\leq n}, \mathsf{e} \cup \mathsf{f}]\, p \vee q, \quad \text{but with} \quad \not\models [\mathsf{U}^{\leq n}, \mathsf{e} \cup \mathsf{f}]p \quad \text{and} \quad \not\models [\mathsf{U}^{\leq n}, \mathsf{e} \cup \mathsf{f}]q$$

Compare, e.g. with Lemma 4.4.3 for the treatment of (valid) disjunctive effects in LCC logics.

Following Chapter 6, non-deterministic actions E_d are constructed from deterministic actions using choice, e.g. $\mathsf{E}_d = \{\mathsf{e}, \mathsf{f}\}$ also denoted $\mathsf{e} \cup \mathsf{f}$. A non-deterministic action with effect $p \vee q$ can be seen as built with an action f_0 for p and an action f_1 for q. (See Section 7.2 below, though, for different types of non-determinism.) We will also make use of the convention $\mathsf{pre}(\mathsf{E}_d) = \bigvee_{\mathsf{e} \in \mathsf{R}_d} \mathsf{pre}(\mathsf{e})$, introduced in Chapter 6.

Example 7.1.1 (Breaking a window)**.** Consider the action of throwing a stone against a window. Suppose the window is composed of two windowpanes made of crystal (left and right). Define $p =$ *the left crystal breaks*, and $q =$ *the right crystal breaks*. If the agent throws the stone against the window, this will instantiate either as $\mathsf{e} =$ *throwing the stone against the left crystal* or $\mathsf{f} =$ *throwing it against the right crystal*. Simplifying a bit, these deterministic actions can be defined as follows

$\mathsf{pre}(\mathsf{e})$	$= \top$	$\mathsf{pre}(\mathsf{f})$	$= \top$
$\mathsf{post}(\mathsf{e})(p)$	$= \top$	$\mathsf{post}(\mathsf{f})(p)$	$= p$
$\mathsf{post}(\mathsf{e})(q)$	$= q$	$\mathsf{post}(\mathsf{f})(q)$	$= \top$
$\mathsf{R}_a(\mathsf{e}, \mathsf{f}')$	iff $\mathsf{f}' = \mathsf{e}$	$\mathsf{R}_a(\mathsf{f}, \mathsf{e}')$	iff $\mathsf{e}' = \mathsf{f}$

The R_a relation is the identity relation because agent a can tell which windowpane has been or is going to be hit by the stone, during (or at the end of) the execution of this action. The action $\mathsf{e} \cup \mathsf{f}$ behaves as above w.r.t. the effects $p \vee q$ and p and q. Note that one of the two crystals remains as it was before the stone was thrown: either the truth-value of p or the truth-value of q is preserved.

The focus of the present chapter is on strong non-deterministic planning. A strong solution (resp. weak solution) for a given planning domain $\mathbb{M} = (\varphi_T, A, \varphi_G)$ is a plan such that all (resp. some) of its possible executions at the initial state φ_T lead to a goal state satisfying φ_G. Thus, in Example 7.1.1, throwing the stone, i.e. $\mathsf{e} \cup \mathsf{f}$, is a strong solution for the planning domain $\mathbb{M} = (\top, \{\mathsf{e} \cup \mathsf{f}\}, \varphi_G)$, if the goal is to break some windowpane $\varphi_G = p \vee q$; and $\mathsf{e} \cup \mathsf{f}$ is a weak solution if the goal is to break the left windowpane $\varphi_G = p$.

For the sake of simplicity, in the present chapter we consider only non-deterministic actions E_d consisting of two deterministic actions $\mathsf{E}_d = \{\mathsf{e}, \mathsf{f}\}$ also denoted with choice as $\mathsf{e} \cup \mathsf{f}$. A generalization of the the present definitions and proofs to the finite case $|\mathsf{E}_d| \geq 2$ can be easily obtained.

Structure of the chapter

In Section 7.2, we first consider a classification of actions in terms of their epistemic properties, related to a similar proposal studied in Bolander et al. [28]. Then in Section 7.3, we propose a planning system for the logics $\mathrm{LCC}_{\cup\otimes n}$. The different refinement steps existing in this planning system are described in Section 7.4, together with a plan search algorithm for the proposed planning system. This section concludes with results for the soundness and completeness of this planning algorithm. Section 7.5 contains the conclusions a description of the related work in the literature.

7.2 Non-determinism and distinguishability

Before proceeding with the introduction of a planning system for $\mathrm{LCC}_{\cup\otimes n}$ logics, let us devote this section to a classification of non-deterministic actions from the point of view of the executing or planner agent. Unless stated otherwise, we will assume in this chapter that these two agents are the same.

For the purpose of planning, the relevant epistemic properties of a non-deterministic action $\mathsf{e} \cup \mathsf{f}$ concern this agent's ability to learn how does $\mathsf{e} \cup \mathsf{f}$ instantiate in the present context: as an instance of e or as an instance of f. In the literature, this has been called run-time distinguishability in [28], when the distinction is made *during the execution of* $\mathsf{e} \cup \mathsf{f}$, as in Example 7.1.1. If the distinction can be made *during the planning phase* (before executing the plan), this action is called plan-time distinguishable. In this section, we study a related classification of non-deterministic actions.

Definition 7.2.1 (Post-action distinguishability)**.** Let $\mathsf{E}_d = \{\mathsf{e}, \mathsf{f}\} \subseteq \mathsf{E}^{\leq n}$ be a non-deterministic action in some action model $\mathsf{U}^{\leq n} = (\mathsf{E}^{\leq n}, \mathsf{R}, \mathsf{pre}, \mathsf{post})$ and let

$a \in \mathsf{Ag}$ be the planner or the executing agent of E_d. We say that E_d is *post-action distinguishable* iff for any pair φ, ψ with $\models [\mathsf{U}^{\leq n}, \mathsf{e}]\varphi \wedge \neg\psi$ and $\models [\mathsf{U}^{\leq n}, \mathsf{f}]\psi \wedge \neg\varphi$ we have that

$$\models [\mathsf{U}^{\leq n}, \mathsf{E}_d] \big([a]\varphi \vee [a]\psi\big)$$

We say that E_d is post-action distinguished at M, w iff $M, w \models [\mathsf{U}^{\leq n}, \mathsf{E}_d] \big([a]\varphi \vee [a]\psi\big)$.

Thus, an agent can distinguish how $\mathsf{E}_d = \mathsf{e} \cup \mathsf{f}$ is being instantiated if and only ifimmediately after the execution of E_d, the agent is aware of the exclusive effects of either e or f. This definition can be trivially satisfied if no such pair φ, ψ exists, as in the action from Example 7.1.1.[1] But for the purpose of the classification, Definition 7.2.1 will suffice.

Below we consider more definitions of distinguishability. Note that, for the sake of simplicity, the definitions below consider ideal notions of (in)distinguishability: some intermediate cases between ideal distinguishability and ideal indistinguishability of the same type are not considered.

Indeed, a natural assumption on the representation of non-deterministic actions E_d, is that the set E_d is closed under the accessibility relation R_a, where a is the executing or planner agent. Moreover, in [28] this relation R_a is assumed to be an equivalence relation. (This is used in [28] to define each action E_d as an equivalence class under R_a.) In this chapter we will also assume this, but (only) with the purpose of obtaining a simple definition of distinguishability during the execution of a non-deterministic action.

Definition 7.2.2 (In-action distinguishability)**.** Let an agent $a \in \mathsf{Ag}$, an action $\mathsf{E}_d \subseteq \mathsf{E}^{\leq n}$ and an action model $\mathsf{U}^{\leq n}$ be as above. We say that E_d is *in-action distinguishable*[2] iff the relation R_a is the identity relation in this set

$$\text{for each } \mathsf{e} \in \mathsf{E}_d \text{ and } \mathsf{f} \in \mathsf{E}^{\leq n} \qquad \mathsf{e}\mathsf{R}_a\mathsf{f} \Leftrightarrow \mathsf{f} = \mathsf{e}$$

We say that E_d is *in-action indistinguishable* iff R_a is the total relation in this set $\mathsf{E}_d \times \mathsf{E}_d \subseteq \mathsf{R}_a$.

Example 7.2.3 (Tossing a coin)**.** Consider the action of tossing a coin (by some agent a), which can result in heads h or tails $\neg h$. The physical part of this action can be modeled with the help of two deterministic actions: *toss heads* and *toss tails*, resp. denoted toss_h $\mathsf{toss}_{\neg h}$ and defined as follows

$\mathsf{pre}(\mathsf{toss}_h)$	$= \top$	$\mathsf{pre}(\mathsf{toss}_{\neg h})$	$= \top$
$\mathsf{post}(\mathsf{toss}_h):$	$h \longmapsto \top$	$\mathsf{post}(\mathsf{toss}_{\neg h}):$	$h \longmapsto \bot$
$(\mathsf{toss}_h, \mathsf{toss}_h)$	$\in \mathsf{R}_a$	$(\mathsf{toss}_{\neg h}, \mathsf{toss}_{\neg h})$	$\in \mathsf{R}_a$

The actions of an *unobservable coin toss* and a *public coin toss* are modeled by extending toss_h into the actions $\mathsf{toss}_h^{\langle \mathsf{Ag} \rangle}$ and $\mathsf{toss}_h^{[\mathsf{Ag}]}$ (and similarly for $\mathsf{toss}_{\neg h}$). As

[1] In this example, the action e of breaking the left crystal says nothing about the status of the right crystal q; and viceversa.

[2] This notion is called run-time distinguishability in [28].

the superindex $\langle \mathsf{Ag} \rangle$ or $[\mathsf{Ag}]$ suggests, in the former case no agent can distinguish *tossing heads* from *tossing tails*. These new actions, then, are defined by the above conditions on pre, post and R_a together with new conditions upon the latter R_a relation.

unobs. coin toss $\mathsf{toss}_h^{\langle \mathsf{Ag} \rangle} \cup \mathsf{toss}_{\neg h}^{\langle \mathsf{Ag} \rangle}$		public coin toss $\mathsf{toss}_h^{[\mathsf{Ag}]} \cup \mathsf{toss}_{\neg h}^{[\mathsf{Ag}]}$	
$(\mathsf{toss}_h^{\langle \mathsf{Ag} \rangle}, \mathsf{toss}_{\neg h}^{\langle \mathsf{Ag} \rangle})$	$\in \mathsf{R}_a$	$(\mathsf{toss}_h^{[\mathsf{Ag}]}, \mathsf{toss}_{\neg h}^{[\mathsf{Ag}]})$	$\notin \mathsf{R}_a$
$(\mathsf{toss}_{\neg h}^{\langle \mathsf{Ag} \rangle}, \mathsf{toss}_h^{\langle \mathsf{Ag} \rangle})$	$\in \mathsf{R}_a$	$(\mathsf{toss}_{\neg h}^{[\mathsf{Ag}]}, \mathsf{toss}_h^{[\mathsf{Ag}]})$	$\notin \mathsf{R}_a$

Note that the deterministic action toss_h (and the same for $\mathsf{toss}_{\neg h}$), although existing in the action model, is not individually available to the agent.

Proposition 7.2.4. *Let $\mathsf{E}_d = \mathsf{e}_0 \cup \mathsf{e}_1 \subseteq \mathsf{E}^{\leq n}$ be an arbitrary non-deterministic action in some action model $\mathsf{U}^{\leq n}$. If E_d is in-action distinguishable, then E_d is post-action distinguishable.*

Proof. For e_0 and φ, we reason as follows:

$$
\begin{array}{lll}
 & \models [\mathsf{U}^{\leq n}, \mathsf{e}_0]\varphi & \\
\Rightarrow & \models [a][\mathsf{U}^{\leq n}, \mathsf{e}_0]\varphi & \text{(by Nec.)} \\
\Leftrightarrow & \models [a] \bigwedge_{\mathsf{e}\mathsf{R}_a\mathsf{f}} [\mathsf{U}^{\leq n}, \mathsf{f}]\varphi & \text{(reduction ax.)} \\
\Leftrightarrow & \models [a][\mathsf{U}^{\leq n}, \mathsf{e}]\varphi & \text{(since } \mathsf{e}\mathsf{R}_a\mathsf{f} \Rightarrow \mathsf{f} = \mathsf{e}\text{)}
\end{array}
$$

From this and a similar reasoning for e_1 and ψ, we conclude that

$$\models [\mathsf{U}^{\leq n}, \mathsf{e}_0]([a]\varphi \vee [a]\psi) \quad \text{and} \quad \models [\mathsf{U}^{\leq n}, \mathsf{e}_1]([a]\varphi \vee [a]\psi)$$

By the reduction axiom for choice, we conclude $\models [\mathsf{U}^{\leq n}, \mathsf{e}_0 \cup \mathsf{e}_1]([a]\varphi \vee [a]\psi)$. □

The random character of the tossing action in Example 7.2.3 is not an essential feature of non-deterministic actions, as the next example shows. This example also illustrates another notion of (in)distinguishability different from Def.7.2.2.

Remark 7.2.5. In the following we add to $\cup$ a new notation $\uplus$ for non-deterministic actions. In terms of the product update semantics, $\mathsf{e} \uplus \mathsf{f}$ is simply a notational variant of $\mathsf{e} \cup \mathsf{f}$. From here on, the expressions $\cup$ and $\uplus$ will just assume different epistemic relations R_a between their components e and f.

In the following example, a truthful conditional announcement about p, has an epistemic precondition $[a]p \vee [a]\neg p$. If this precondition is true, then intuitively the agent can distinguish, before executing the (planned) action, whether it is an announcement that p or an announcement that $\neg p$.

Example 7.2.6. Different announcements that p or about p can be classified in terms of the agent's knowledge of their (actual) properties. In contrast, a truthful public announcement $p!^a_{\mathsf{Ag}}$ really denotes a public announcement that p

which happens to be truthful, since we merely have $\mathsf{pre}(p!^{a}_{\mathsf{Ag}}) = p$. A *consciously (resp. sincerely) truthful announcement that p* requires the agent to know (resp. believe) that p. Let us denote this as $p!^{[a]}_{\mathsf{Ag}}$; and analogously for conscious (resp. intentional) lies, denoted $p\dagger^{[a]}_{\mathsf{Ag}}$. Their preconditions are

$$\mathsf{pre}(p!^{[a]}_{\mathsf{Ag}}) = [a]\mathsf{pre}(p!^{a}_{\mathsf{Ag}}) = [a]p \qquad \mathsf{pre}(p\dagger^{[a]}_{\mathsf{Ag}}) = [a]\mathsf{pre}(p\dagger^{a}_{\mathsf{Ag}}) = [a]\neg p$$

As usual, their postconditions are trivial: $\mathsf{post}(\cdot)(q) = q$, for any $q \in \mathsf{Var}$. From these deterministic actions, the following actions can be defined as well.

- a *careless* announcement that p, made irregardless of its truth-value or the agent's knowledge about it, denoted $p!^{a}_{\mathsf{Ag}} \cup p\dagger^{a}_{\mathsf{Ag}}$
- a *possibly lying* announcement that p, denoted $p!^{[a]}_{\mathsf{Ag}} \underline{\cup} p\dagger^{[a]}_{\mathsf{Ag}}$
- a *sincere* announcement *about* p, denoted $p!^{[a]}_{\mathsf{Ag}} \uplus \neg p!^{[a]}_{\mathsf{Ag}}$
- a *consciously ignorant* announcement that p, denoted $p!^{\langle a\rangle}_{\mathsf{Ag}} \cup p\dagger^{\langle a\rangle}_{\mathsf{Ag}}$

These actions are assigned the preconditions according to the convention $\mathsf{pre}(\mathsf{e} \cup \mathsf{f}) = \mathsf{pre}(\mathsf{e}) \vee \mathsf{pre}(\mathsf{f}) = \mathsf{pre}(\mathsf{e} \uplus \mathsf{f})$. The relations R_a for all these actions are reflexive, and moreover, we assign

$$(\mathsf{e},\mathsf{f}), (\mathsf{f},\mathsf{e}) \in \mathsf{R}_a(\mathsf{e} \cup \mathsf{f}), \mathsf{R}_a(\mathsf{e} \underline{\cup} \mathsf{f}) \qquad \text{and} \qquad (\mathsf{e},\mathsf{f}), (\mathsf{f},\mathsf{e}) \notin \mathsf{R}_a(\mathsf{e} \uplus \mathsf{f})$$

As some of these examples show, the preconditions of a planned non-deterministic action can also contribute to the distinguishability between its components. In this case, the distinction is made just before the execution of this action, not during it.

Definition 7.2.7 (Pre-action distinguishability)**.** Let an agent $a \in \mathsf{Ag}$, an action $\mathsf{E}_d \subseteq \mathsf{E}^{\leq n}$ and an action model $\mathsf{U}^{\leq n}$ be as above. We say that E_d is *pre-action distinguishable* iff the preconditions are logically independent, in the following sense

$$\not\models \mathsf{pre}(\mathsf{e}) \to \mathsf{pre}(\mathsf{f}) \quad \text{and} \quad \not\models \mathsf{pre}(\mathsf{f}) \to \mathsf{pre}(\mathsf{e})$$

We say that E_d is pre-action indistinguishable iff $\mathsf{pre}(\mathsf{e}) \equiv \mathsf{pre}(\mathsf{f})$.

In practice, that the actions $\mathsf{e}, \mathsf{f} \in \mathsf{E}_d$ can be (pre-action) distinguished in the context M, w where E_d is to be executed will depend on the information the agent has about this particular context.

Proposition 7.2.8. *Let $\mathsf{E}_d = \mathsf{e}_0 \cup \mathsf{e}_1 \subseteq \mathsf{E}^{\leq n}$ be an arbitrary non-deterministic action in some action model $\mathsf{U}^{\leq n}$. Assume that E_d is pre-action distinguishable. If $M, w \models [a](\mathsf{pre}(\mathsf{e}_0) \wedge \neg\mathsf{pre}(\mathsf{e}_1))$ then E_d is post-action distinguished at M, w.*

Proof. We reason as follows:

$$
\begin{array}{lll}
M,w \models & [a](\mathsf{pre}(\mathsf{e}_0) \wedge \neg \mathsf{pre}(\mathsf{e}_1)) & \\
M,w \models & [a]\mathsf{pre}(\mathsf{e}_0) \wedge [a]\neg \mathsf{pre}(\mathsf{e}_1) & \\
M,w \models & [a]\mathsf{pre}(\mathsf{e}_0) \wedge [a]\mathsf{pre}(\mathsf{e}_1) \to [\mathsf{U}^{\leq n}, \mathsf{e}_1]\varphi & \\
M,w \models & [a]\mathsf{pre}(\mathsf{e}_0) \wedge [a][\mathsf{U}^{\leq n}, \mathsf{e}_1]\varphi & \text{(by Lemma 4.4.3)} \\
M,w \models & [a][\mathsf{U}, \mathsf{e}_0]\varphi \wedge [a][\mathsf{U}^{\leq n}, \mathsf{e}_1]\varphi & \text{(by Nec. on assumption)} \\
M,w \models & [a] \bigwedge_{\mathsf{e}_0 \mathsf{R}_a \mathsf{f}} [\mathsf{U}^{\leq n}, \mathsf{f}]\varphi \wedge & \\
 & [a] \bigwedge_{\mathsf{e}_1 \mathsf{R}_a \mathsf{f}} [\mathsf{U}^{\leq n}, \mathsf{f}]\varphi & \text{(by def. } \mathsf{R}_a\text{)} \\
M,w \models & [\mathsf{U}^{\leq n}, \mathsf{e}_0][a]\varphi \wedge [\mathsf{U}^{\leq n}, \mathsf{e}_1][a]\varphi & \text{(by Reduction Axiom for } [a]\text{)} \\
M,w \models & [\mathsf{U}^{\leq n}, \mathsf{E}_d][a]\varphi & \text{(by Reduction Axiom for } \cup\text{)} \\
M,w \models & [\mathsf{U}^{\leq n}, \mathsf{E}_d]([a]\varphi \vee [a]\psi) &
\end{array}
$$

□

Propositions 7.2.4 and 7.2.8 show two ways for an execution of $\mathsf{e} \cup \mathsf{f}$ to be (post-action) distinguished as e or as f. Moreover, it can be shown that these two are the only possible ways for this agent to distinguish between e and f.

Proposition 7.2.9. *Assume $M, w \models \langle a \rangle \mathsf{pre}(\mathsf{E}_d)$. Then, if E_d is pre-action indistinguishable and in-action indistinguishable, then E_d is not post-action distinguished at M, w.*

Proof. Let $\mathsf{E}_d = \mathsf{e}_0 \cup \mathsf{e}_1$, and let φ, ψ be arbitrary formulas satisfying $\models [\mathsf{U}^{\leq n}, \mathsf{e}_0]\varphi \wedge \neg\psi$ and $\models [\mathsf{U}^{\leq n}, \mathsf{e}_1]\neg\varphi \wedge \psi$. Note that, by (Nec), it is also valid that $\models [a][\mathsf{U}^{\leq n}, \mathsf{e}_0]\varphi \wedge \neg\psi$, and so on. Now we can reason according to the following equivalences. On the one hand,

$$
\begin{array}{lll}
M,w \models & \langle a \rangle \mathsf{pre}(\mathsf{E}_d) & \\
M,w \models & \langle a \rangle (\mathsf{pre}(\mathsf{e}_0) \vee \mathsf{pre}(\mathsf{e}_1)) & \text{by def. of } \mathsf{pre}(\mathsf{E}_d) \\
M,w \models & \langle a \rangle \mathsf{pre}(\mathsf{e}_1) & \text{by assumption } \mathsf{pre}(\mathsf{e}_0) \equiv \mathsf{pre}(\mathsf{e}_1) \\
M,w \models & \langle a \rangle (\langle \mathsf{U}^{\leq n}, \mathsf{e}_1 \rangle \top \wedge [\mathsf{U}^{\leq n}, \mathsf{e}]\neg\varphi) & \text{by Lemma 4.4.3; and (Nec)} \\
M,w \models & \langle a \rangle \langle \mathsf{U}^{\leq n}, \mathsf{e}_1 \rangle \neg\varphi & \text{by Lemma 4.4.3} \\
M,w \not\models & [a][\mathsf{U}^{\leq n}, \mathsf{e}_1]\varphi & \\
M,w \not\models & [a] \bigwedge_{\mathsf{e}_0 \mathsf{R}_a \mathsf{f}} [\mathsf{U}^{\leq n}, \mathsf{f}]\varphi & \text{since } \mathsf{e}_0 \mathsf{R}_a \mathsf{e}_1 \\
M,w \not\models & [\mathsf{U}^{\leq n}, \mathsf{e}_0][a]\varphi &
\end{array}
$$

On the other hand, we also have that $M, w \not\models [\mathsf{U}^{\leq n}, \mathsf{e}_0][a]\psi$. To show this, assume the contrary towards a contradiction

$$
\begin{array}{lll}
M,w \models & [\mathsf{U}^{\leq n}, \mathsf{e}_0][a]\psi & \text{assumption} \\
M,w \models & [a] \bigwedge_{\mathsf{e}_0 \mathsf{R}_a \mathsf{f}} [\mathsf{U}^{\leq n}, \mathsf{f}]\psi & \text{reduction axiom for } \pi = a \\
M,w \models & [a][\mathsf{U}^{\leq n}, \mathsf{e}_0]\psi & \text{since } \mathsf{e}_0 \mathsf{R}_a \mathsf{e}_0 \\
M,w \models & [a][\mathsf{U}^{\leq n}, \mathsf{e}_0]\neg\psi & \text{applying (Nec) to } \models [\mathsf{U}^{\leq n}, \mathsf{e}_0]\neg\psi \\
M,w \models & [a][\mathsf{U}^{\leq n}, \mathsf{e}_0]\bot & \text{by the last two claims} \\
M,w \models & [a]\neg \mathsf{pre}(\mathsf{e}_0) & \text{by Lemma 4.4.3}
\end{array}
$$

	pre-action indist.	pre-action dist.
in-action indist.	$\cup$ unobs. coin toss, ignorant ann. that p	$\underline{\cup}$ ignorant switch, careless ann. that p
in-action dist.	$\uplus$ public coin toss	$\uplus$ conscious switch, conscious ann. about p

The last claim contradicts the fact that $M, w \models \langle a \rangle \mathsf{pre}(\mathsf{e}_0)$, which obtains from the initial assumptions $M, w \models \langle a \rangle(\mathsf{pre}(\mathsf{e}_0) \vee \mathsf{pre}(\mathsf{e}_1))$ and $\mathsf{pre}(\mathsf{e}_0) \equiv \mathsf{pre}(\mathsf{e}_1)$. Thus, we conclude that $M, w \not\models [\mathsf{U}^{\leq n}, \mathsf{e}_0][a]\psi$.

From this and the previous claim shown above, we conclude that

$$M, w \not\models [\mathsf{U}^{\leq n}, \mathsf{e}_0]([a]\varphi \vee [a]\psi)$$

Finally, by the reduction axiom for choice, this suffices to show that $M, w \not\models [\mathsf{U}^{\leq n}, \mathsf{e}_0 \cup \mathsf{e}_1]([a]\varphi \vee [a]\psi)$, so we are done. □

For an example of a pre-action distinguishable action which is not purely epistemic (in contrast to Example 7.2.6), consider the following.

Example 7.2.10 (Switching the light)**.** Pressing a button can switch the light on or off, causing it to be resp. in state *on* or $\neg on$. Let us denote these as actions on and off. Their (physical) preconditions and effects are

$$\mathsf{pre}(\mathsf{on}) = \neg on \quad \mathsf{post}(\mathsf{on})(on) = \top \qquad \mathsf{pre}(\mathsf{off}) = on \quad \mathsf{post}(\mathsf{off})(on) = \bot$$

Similarly to Example 7.2.6, one can define the actions of a *conscious* or an *ignorant switching*, e.g. by an agent who is able to see, or resp., who enters the room blindfolded agent.

Propositions 7.2.4–7.2.9 suggest to classify non-deterministic actions according to their in-action or pre-action (in)distinguishability. See the next table for examples of the different types of actions. This table also contains the notation for each type of non-deterministic action, namely $\cup, \underline{\cup}, \uplus$, to be used in the next section.

The different notation $\cup, \uplus, \underline{\cup}$ are notational variants just used to classify the actions E_d during a planning task. The semantics of $[\mathsf{U}, \mathsf{e} \cup \mathsf{f}]$, $[\mathsf{U}, \mathsf{e} \uplus \mathsf{f}]$ and $[\mathsf{U}, \mathsf{e} \underline{\cup} \mathsf{f}]$ are exactly the same as defined in the previous chapter for choice $[\mathsf{U}, \mathsf{e} \cup \mathsf{f}]$.

We can proceed to the study of planning in the $\mathrm{LCC}_{\cup \otimes n}$ logics. As the examples in this section show, the notions of *available action*, *plan* and *solution* from Chapter 5 must be redefined to the present non-deterministic case.

7.3 A non-deterministic planning system for $\mathrm{LCC}_{\cup\otimes n}$ logics.

As mentioned before, from here on, we will abstract from any particular bound n upon the length of composite actions in plans, so in the following we will just write the action model as U rather than as a fixed action model $\mathsf{U}^{\leq n}$. With this remark in mind, recall that the *courses of actions* definable in some $\mathrm{LCC}_{\cup\otimes n}$ logic, for a given action model U, are sequences of the form

$$[\mathsf{U},\mathsf{E}_1]\dots[\mathsf{U},\mathsf{E}_k]$$

where each $\mathsf{E}_i = (\mathsf{e}_0 \otimes \cdots \otimes \mathsf{e}_k) \cup (\mathsf{f}_0 \otimes \cdots \otimes \mathsf{f}_{k'})$. Sequences of this form will also be written as $(\mathsf{E}_0, \dots, \mathsf{E}_k)$ during the planning phase.

Not all of these sequences of modalities in the language, though, represent executable plans, in the sense that some combinations demand unrealistic conditions upon the knowledge of the executing agent.

Example 7.3.1 (Tossing a coin; Cont'd)**.** A *coin toss with unobserved result*, followed by a *conscious truthful announcement of this result* is in practice not executable; that is, the action

$$[\mathsf{U},\mathsf{toss}_h \cup \mathsf{toss}_{\neg h}][\mathsf{U}, h!^{[a]}_{\mathsf{Ag}} \cup \neg h!^{[a]}_{\mathsf{Ag}}]$$

The reason is that after the coin toss, the agent or nature cannot randomly choose between announcements $h!$ and $\neg h!$, if these are to be truthful. Combinations like these, then, should not be allowed in the construction of plans. From a more formal point of view, the problem with this example can be grasped as follows. Set our goal to be the precondition of the announcement action, i.e. $\varphi_{\mathsf{goals}(h!\cup\neg h!)} = \mathsf{pre}(h!^{a}_{\mathsf{Ag}} \cup \neg h!^{a}_{\mathsf{Ag}}) = [a]h \vee [a]\neg h$. Now, computing this goal as we did in Chapter 5 (but using the translation function t for $\mathrm{LCC}_{\cup\oplus n}$) gives $\varphi_{\mathsf{goals}(h!\cup\neg h!,\mathsf{toss})} = \langle a\rangle\bot$, so this (non-sense) plan requires that the agent starts with inconsistent beliefs.

The space of plans, then, must be a proper subset of the set of sequences, due to these examples and again to the fact that the available deterministic actions A_{E} is not the whole set E. The first step is to define the planning domains that add non-deterministic actions to the deterministic planning domains of Definition 5.2.1.

Definition 7.3.2 (Planning domain)**.** For a given action model U, a triple

$$\mathbb{M} = (\varphi_T, A, \varphi_G)$$

is a *non-deterministic planning domain* for agent a in U iff φ_T, φ_G are consistent $\mathcal{L}_{\mathsf{U}}$-formulas without action modalities, and $A = A_{\mathsf{E}} \cup A_{\cup} \cup A_{\uplus} \cup A_{\underline{\cup}}$ is the set of actions available to some agent a, where

- $A_{\mathsf{E}} \subseteq \mathsf{E}$ is a set of deterministic actions,

- $A_{\cup} \subseteq \mathsf{E} \times \mathsf{E}$ is a set of non-deterministic actions $\mathsf{E}_d = \mathsf{e} \cup \mathsf{f}$, such that $\mathsf{e} \cup \mathsf{f}$ is in-action indistinguishable and pre-action indistinguishable
- $A_{\uplus} \subseteq \mathsf{E} \times \mathsf{E}$ is a set of in-action distinguishable actions
- $A_{\underline{\cup}} \subseteq \mathsf{E} \times \mathsf{E}$ is a set of pre-action distinguishable actions, which are not in-action distinguishable

Example 7.3.3. [Tossing a coin, Cont'd] Recall the *unobserved coin toss* action from Example7.2.3. Let us read this action $\mathsf{toss}_h \cup \mathsf{toss}_{\neg h}$ as a coin toss that ends with the coin in the palm of agent a's hand, where it remains unobserved. Assume this agent can execute a sensing action, *feeling whether the coin in one's hand landed heads*; and a *(conscious) flip into heads* action, defined as follows:

feel_h		$\mathsf{feel}_{\neg h}$		$\mathsf{flip}_h^{[a]}$	
$\mathsf{pre}(\mathsf{feel}_h)$	$= h$	$\mathsf{pre}(\mathsf{feel}_{\neg h})$	$= \neg h$	$\mathsf{pre}(\mathsf{flip}_h^{[a]})$	$= [a]\neg h$
$\mathsf{post}(\mathsf{feel}_h)(q)$	$= q$	$\mathsf{post}(\mathsf{feel}_{\neg h})(q)$	$= q$	$\mathsf{post}(\mathsf{flip}_h^{[a]})(h)$	$= \top$
				$\mathsf{post}(\mathsf{flip}_h^{[a]})(q)$	$= q,\ \forall q \neq h$
$(\mathsf{feel}_h, \mathsf{feel}_h)$	$\in \mathsf{R}_a$	$(\mathsf{feel}_{\neg h}, \mathsf{feel}_{\neg h})$	$\in \mathsf{R}_a$	$(\mathsf{flip}_h^{[a]}, \mathsf{flip}_h^{[a]})$	$\in \mathsf{R}_a$
$(\mathsf{feel}_h, \mathsf{feel}_{\neg h})$	$\notin \mathsf{R}_a$	$(\mathsf{feel}_{\neg h}, \mathsf{feel}_h)$	$\notin \mathsf{R}_a$	$(\mathsf{flip}_h^{[a]}, \mathsf{f})$	$\notin \mathsf{R}_a$

Using these elements, the previous actions would be represented by the following sets:

$$\begin{array}{lcllcl} \mathsf{E} & = & \{\mathsf{toss}_h^{\langle \mathsf{Ag} \rangle}, \mathsf{toss}_{\neg h}^{\langle \mathsf{Ag} \rangle}, \mathsf{feel}_h, \mathsf{feel}_h, \mathsf{flip}_h\} & & & \\ A_{\mathsf{E}} & = & \{\mathsf{flip}_h^{[a]}\} & A_{\cup} & = & \{\mathsf{toss}_h^{\langle \mathsf{Ag} \rangle} \cup \mathsf{toss}_{\neg h}^{\langle \mathsf{Ag} \rangle}\} \\ A_{\uplus} & = & \{\mathsf{feel}_h \uplus \mathsf{feel}_{\neg h}\} & A_{\underline{\cup}} & = & \varnothing \end{array}$$

A planning domain $\mathbb{M} = (\varphi_T, A, \varphi_G)$ determines which combinations of actions from A, called $\mathbb{M}$-actions, are considered as executable by the planner agent. Each of these $\mathbb{M}$-actions include arbitrary composite actions and a single choice operator, and are inductively defined as follows.

Definition 7.3.4 ($\mathbb{M}$-action; $\mathbb{M}$-sequence)**.** We say $[\mathsf{U}, \mathsf{e}]$ and $[\mathsf{U}, \mathsf{e} \cup \mathsf{f}]$ are $\mathbb{M}$-sequences whenever $\mathsf{e} \in A_{\mathsf{E}}$ and $\mathsf{e} \cup \mathsf{f}$ in $A_{\cup}$. Moreover, if $\mathsf{e}' \otimes \cdots \otimes \mathsf{e}''$ and $\mathsf{f}' \otimes \cdots \otimes \mathsf{f}''$ are elements of $A_{\mathsf{E}}^{<\omega}$ and $\mathsf{e} \cup \mathsf{f}$ is of the form $\mathsf{e} \uplus \mathsf{f} \in A_{\uplus}$ or $\mathsf{e} \underline{\cup} \mathsf{f} \in A_{\underline{\cup}}$, then

$$[\mathsf{U}, (\mathsf{e} \otimes \mathsf{e}' \otimes \cdots \otimes \mathsf{e}'') \cup (\mathsf{f} \otimes \mathsf{f}' \otimes \cdots \otimes \mathsf{f}'')] \quad \text{is an } \mathbb{M}\text{-action}$$

Finally, any finite sequence $[\mathsf{U}, \mathsf{E}_k] \ldots [\mathsf{U}, \mathsf{E}_1]$ of $\mathbb{M}$-actions is called an $\mathbb{M}$-*sequence*.

Example 7.3.5. [Tossing a coin; Cont'd] Using the actions defined in Example 7.3.3, the following is an $\mathbb{M}$-sequence

$$[\mathsf{U}, \mathsf{toss}_h^{\langle \mathsf{Ag} \rangle} \cup \mathsf{toss}_{\neg h}^{\langle \mathsf{Ag} \rangle}] \quad [\mathsf{U}, (\mathsf{feel}_h \uplus (\mathsf{feel}_{\neg h} \otimes \mathsf{flip}_h)]$$
tossing the coin, sensing it, and if *tails* flip it to *heads*

Definition 7.3.6 (Strong solution)**.** We say that an $\mathbb{M}$-sequence $[\mathsf{U}, \mathsf{E}_1], \ldots, [\mathsf{U}, \mathsf{E}_r]$ is a *strong solution* for a planning domain $\mathbb{M} = (\varphi_T, A, \varphi_G)$ iff

$$\models \varphi_T \to [\mathsf{U}, \mathsf{E}_1] \ldots [\mathsf{U}, \mathsf{E}_r] \varphi_G \qquad (\textit{success})$$
$$\models \varphi_T \to \langle \mathsf{U}, \mathsf{E}_1 \rangle \ldots \langle \mathsf{U}, \mathsf{E}_r \rangle \top \qquad (\textit{executability})$$

From here on, we will refer to strong solutions simply as solutions. It can be shown that the $\mathbb{M}$-sequence from Example 7.3.5 is a solution for the planning domain

$$\mathbb{M} = (\ \top,\ \{\ \mathsf{toss}_h^{\langle \mathsf{Ag} \rangle} \cup \mathsf{toss}_{\neg h}^{\langle \mathsf{Ag} \rangle},\ \mathsf{feel}_h \uplus \mathsf{feel}_{\neg h},\ \mathsf{flip}_h^{[a]}\ \},\ [a]h)$$

7.4 A Search Algorithm for Non-deterministic Plans

As in Chapter 5, we study a Breadth First Search algorithm for planning in $\mathrm{LCC}_{\cup \otimes n}$ logics. As usual, the planning algorithm will search within the space of plans for some construction satisfying the Terminating Condition. In contrast to the algorithm for the deterministic case, though, not only the plans, but also some of the actions $[\mathsf{U}, \mathsf{E}_d]$ are built (during the plan construction) in a stepwise fashion. The actions under construction are called partial actions.

Definition 7.4.1. [Partial action] Given an action model U, a *partial action* in U is an expression of the form $\mathsf{e} \cup \mathsf{f}$ where

$$\mathsf{f} = (\mathsf{f}_0 \otimes \mathbf{x} \otimes \mathsf{f}_1) \qquad \text{or} \qquad \mathsf{f} = (\mathsf{f}_0 \otimes \mathbf{x})$$

and satisfying $\mathsf{e}_0 \uplus \mathsf{f}_0 \in A_{\uplus}$ or $\mathsf{e}_0 \underline{\cup} \mathsf{f}_0 \in A_{\underline{\cup}}$.

As a consequence of this, not all plans considered during search correspond to $\mathbb{M}$-sequence of modalities in the logic of some $\mathsf{U}^{\leq n}$ action model.

The structure of a plan is again a pair consisting of: a formula for the open goals, and a (possibly empty) $\mathbb{M}$-sequence $\pi_k = [\mathsf{U}, \mathsf{E}_k] \ldots [\mathsf{U}, \mathsf{E}_1]$ prefixed by a (possibly empty) partial action.

$$\begin{array}{rcccc} \text{plan} = & (& \text{partial act.} + \mathbb{M}\text{-sequence}, & \text{open goals} &) \\ \pi \ = & (& [\mathsf{U}, \mathsf{e} \cup \mathsf{f}]\pi_k, & \varphi_{\mathsf{goals}(\pi)} &) \end{array}$$

Note we will abuse notation: when a plan π_k simply consists of an $\mathbb{M}$-sequence $[\mathsf{U}, \mathsf{E}_k], \ldots, [\mathsf{U}, \mathsf{E}_1]$ we will refer to this $\mathbb{M}$-sequence again as π_k. See also Figure 7.1 for an illustration of Definitions 7.4.2–7.4.4.

Definition 7.4.2 (Empty plan; Refinement with $A_{\cup}$)**.** Let $\mathbb{M} = (\varphi_T, A, \varphi_G)$ be a planning domain. We define the *empty plan* as the pair

$$\pi_{\varnothing} = (\varnothing, \varphi_G)$$

Let $\pi_k = [\mathsf{U}, \mathsf{E}_k] \dots [\mathsf{U}, \mathsf{E}_1]$ be an $\mathbb{M}$-sequence, and let $\mathsf{e}_0 \cup \mathsf{f}_0 \in A_\cup$. We define the refinement $\pi_k(\mathsf{e}_0 \cup \mathsf{f}_0)$ as the pair

$$([\mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0]\pi_k,\ \varphi_{\mathsf{goals}(\pi_k(\mathsf{e}_0 \cup \mathsf{f}_0))})$$

where $\varphi_{\mathsf{goals}(\pi_k(\mathsf{e}_0 \cup \mathsf{f}_0))}) = t([\mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0]\mathsf{goals}(\pi_k) \wedge \langle \mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0 \rangle \top)$.

Definition 7.4.3 (Refinement with $A_\uplus$ or $A_\cup$)**.** Let $\mathbb{M}$ be a planning domain and $\pi_k = [\mathsf{U}, \mathsf{E}_k][\mathsf{U}, \mathsf{E}_{k-1}] \dots [\mathsf{U}, \mathsf{E}_1]$ an arbitrary $\mathbb{M}$-sequence, with $\mathsf{E}_k \in A_\mathsf{E}^{<\omega}$ deterministic. For any action $\mathsf{e}_0 \cup \mathsf{f}_0 \in A_\uplus \cup A_\cup$, i.e. with $\cup \in \{\uplus, \cup\}$, we define the refinement $\pi_k(\mathsf{e}_0 \uplus \mathsf{f}_0)$ as either of the following pairs, also denoted resp. $\pi_\varnothing(\dots, \mathsf{e}_0 \cup \mathsf{f})$ and $\pi_\varnothing(\dots, \mathsf{e}_0 \cup \mathsf{f}_0;)$,

$$([\mathsf{U}, (\mathsf{e}_0 \otimes \mathsf{E}_k) \cup (\mathsf{f}_0 \otimes \mathbf{x})]\pi_{k-1}, \varphi_{\mathsf{goals}(\pi_{k-1})})$$
$$([\mathsf{U}, (\mathsf{e}_0 \otimes \mathsf{E}_k) \cup \mathsf{f}_0]\pi_{k-1}, t([\mathsf{U}, (\mathsf{e}_0 \otimes \mathsf{E}_k) \cup \mathsf{f}_0]\varphi_{\mathsf{goals}(\pi_{k-1})} \wedge \langle \mathsf{U}, (\mathsf{e}_0 \otimes \mathsf{E}_k) \cup \mathsf{f}_0 \rangle \top))$$

Definition 7.4.4 (Refinement with A_E)**.** Let $\mathbb{M}$ be a planning domain and let $\pi_k = [\mathsf{U}, \mathsf{E}_k][\mathsf{U}, \mathsf{E}_{k-1}] \dots [\mathsf{U}, \mathsf{E}_1]$ be a plan where $[\mathsf{U}, \mathsf{E}_k$ is an $\mathbb{M}$-action or a partial action and $[\mathsf{U}, \mathsf{E}_{k-1}] \dots [\mathsf{U}, \mathsf{E}_1]$ is an arbitrary $\mathbb{M}$-sequence. Let $\mathsf{e} \in A_\mathsf{E}$ be a deterministic action. Define $\varphi^*_{\mathsf{goals}(\pi_k(\mathsf{e}))} = t([\mathsf{U}, \mathsf{e}]\varphi_{\mathsf{goals}(\pi_k)} \wedge \langle \mathsf{U}, \mathsf{e} \rangle \top)$.

Let $\mathsf{E}_k \in A_\mathsf{E}^{<\omega}$ be a deterministic $\mathbb{M}$-action. Define the refinement $\pi_k(\mathsf{e})$ as

$$([\mathsf{U}, \mathsf{e} \otimes \mathsf{E}_k]\pi_{k-1}, \varphi^*_{\mathsf{goals}(\pi_k(\mathsf{e}))})$$

Let $\mathsf{E}_k \notin A_\mathsf{E}^{<\omega}$ be a non-deterministic $\mathbb{M}$-action. We define $\pi_k(\mathsf{e})$ as

$$([\mathsf{U}, \mathsf{e}]\pi_k, \varphi^*_{\mathsf{goals}(\pi_k(\mathsf{e}))})$$

Let E_k be a partial action $\mathsf{E}_k = \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathbf{x} \otimes \mathsf{f}_1)$ with $\cup \in \{\uplus, \cup\}$ and $\mathsf{e}' = (\mathsf{e}'_0 \otimes \cdots)$.

We define the refinement $\pi_\varnothing(\dots, \mathsf{e})$, denoted $\pi_k(\mathsf{e})$, as

$$([\mathsf{U}, \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathbf{x} \otimes \mathsf{e} \otimes \mathsf{f}_1)], \varphi^*_{\mathsf{goals}(\pi_k(\mathsf{e}))})$$

We also define the refinement $\pi_\varnothing(\dots, \mathsf{e};)$, also denoted $\pi_k(\mathsf{e})$, as

$$([\mathsf{U}, \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathsf{e} \otimes \mathsf{f}_1)], \varphi_{\mathsf{goals}(\pi_k(\mathsf{e}))})$$

where $\varphi_{\mathsf{goals}(\pi_k(\mathsf{e}))} =$

$$\begin{cases} t\left([\mathsf{U}, \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathsf{e} \otimes \mathsf{f}_1)]\varphi_{\mathsf{goals}(\pi_{k-1})} \wedge \langle \mathsf{U}, \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathsf{e} \otimes \mathsf{f}_1) \rangle \top\right) & \text{if } \cup = \uplus \\ \Big([a](\mathsf{pre}(\mathsf{e}'_0) \wedge \neg \mathsf{pre}(\mathsf{f}_0)) \vee [a](\neg \mathsf{pre}(\mathsf{e}'_0) \wedge \mathsf{pre}(\mathsf{f}_0))\Big) \wedge \\ t\left([\mathsf{U}, \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathsf{e} \otimes \mathsf{f}_1)]\varphi_{\mathsf{goals}(\pi_{k-1})} \wedge \langle \mathsf{U}, \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathsf{e} \otimes \mathsf{f}_1) \rangle \top\right) & \text{if } \cup = \cup \end{cases}$$

The refinements with e of plans containing partial actions of the form $\mathsf{E}_k = \mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathbf{x})$ are analogously defined for $\cup \in \{\uplus, \cup\}$, as $\mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathbf{x} \otimes \mathsf{e})$ or $\mathsf{e}' \cup (\mathsf{f}_0 \otimes \mathsf{e})$ and the corresponding goals $\varphi_{\mathsf{goals}(\pi_k(\mathsf{e}))}$ defined as above.

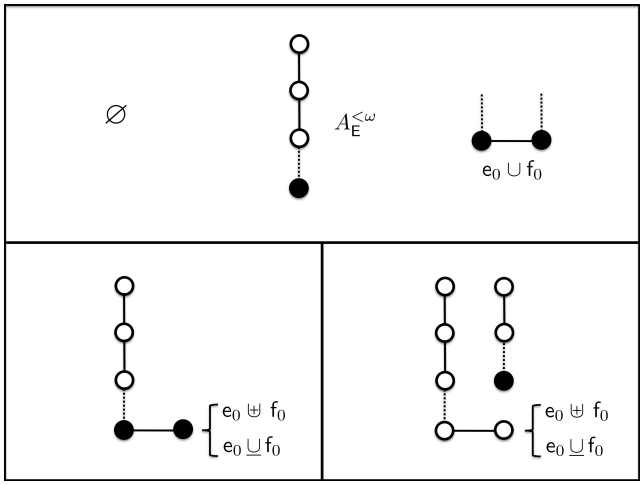


Figure 7.1: This figure represents refinements (black circles) used for the construction of actions (Top) Some types of actions built during the construction of a non-deterministic plan. (Bottom) Partial actions built introducing a pre- or in-action dist. action; or by refining the latter with a deterministic action (right).

Thus, a plan can be denoted as an $\mathbb{M}$-sequence, or as a sequence of plan refinements from the empty plan

$$[\mathsf{U}, \mathsf{E}_k] \dots [\mathsf{U}, \mathsf{E}_1] \qquad \text{or} \qquad \pi_\varnothing(\mathsf{f}, \dots, \mathsf{e}_0 \cup \mathsf{f}_0, \dots, \mathsf{f}')$$

Definition 7.4.5 (Plan. Leaf plan)**.** For a given planning domain $\mathbb{M}$, a *plan* for $\mathbb{M}$ is any pair $(\pi, \varphi_{\mathsf{goals}(\pi)})$ obtained after a finite number of applications of Definitions 7.4.2-7.4.4 upon the empty plan $\pi_\varnothing$ for $\mathbb{M}$.

Given a plan π for $\mathbb{M}$ and a refinement $\pi(\mathsf{e})$ of π with a deterministic action $\mathsf{e} \in A_{\mathsf{E}}$, we say that $\pi(\mathsf{e})$ is a *leaf* iff $\varphi_{\mathsf{goals}(\pi(\mathsf{e}))}$ is inconsistent or $\models \varphi_{\mathsf{goals}(\pi(\mathsf{e}))} \to \varphi_{\mathsf{goals}(\pi)}$.

Given a non-deterministic $\mathbb{M}$-action E_k, we say $\pi_k = [\mathsf{U}, \mathsf{E}_k] \dots [\mathsf{U}, \mathsf{E}_1]$ is a *leaf* iff $\varphi_{\mathsf{goals}(\pi_k)}$ is inconsistent.

Definition 7.4.6 (Terminating Condition)**.** The Terminating Condition for a plan π for $\mathbb{M}$ to be the output of a planning algorithm is $\models \varphi_T \to \varphi_{\mathsf{goals}(\pi)}$.

As usual, the construction of plans will stop if the current plan's open goals follow from the initial state φ_T.

Data: $\mathbb{M} = (\varphi_T, A, \varphi_G)$
Result: π; or fail
initialization: $\pi = \pi_\varnothing$ and $\mathsf{Plans} = \langle \pi \rangle$;
while $\not\models \varphi_T \to \varphi_{\mathsf{goals}(\pi)}$ **do**
  delete π from Plans;
  set $\mathsf{Plans} = \mathsf{Plans}^\frown\langle\, \pi' \mid \pi'$ is a refinement of π and π' is not a leaf $\rangle$;
  if $\mathsf{Plans} = \varnothing$ **then**
    set $\pi = \mathsf{fail}$
  else
    set $\pi =$ the first element of Plans
  end
end

Algorithm 5: Breadth First Search for backward non-deterministic planning in $\mathrm{LCC}_{\cup\otimes n}$.

Let us finally address the properties of this algorithm for non-deterministic planning based on Breadth First Search in the space of plans.

Theorem 7.4.7 (Soundness)**.** *Let π be the output of the BFS algorithm in Fig. 5 for an arbitrary planning domain $\mathbb{M}$. Then, π is a solution for $\mathbb{M}$.*

Proof. Clearly, π is an expression of the form $\pi_\varnothing(\ldots)$ for some sequence of actions $\ldots$ in $A^{<\omega}$. First we show by induction that π is an $\mathbb{M}$-sequence, say of the form $[\mathsf{U}, \mathsf{E}_1] \ldots [\mathsf{U}, \mathsf{E}_m]$.

(Base Case m) We show that some $[\mathsf{U}, \mathsf{E}_m]$ is defined by an initial fragment of the output $\pi = \pi_\varnothing(\ldots)$ as an $\mathbb{M}$-action. Let then $\pi = \pi_\varnothing(\ldots, \mathsf{e}_0 \cup \mathsf{f}_0, \ldots)$ be such that $\mathsf{e}_0 \cup \mathsf{f}_0$ is the first action occurring in π that is not in A_E, i.e. $\pi = \pi_\varnothing(\mathsf{e}_i, \ldots, \mathsf{e}_{i'}, \mathsf{e}_0 \cup \mathsf{f}_0, \ldots)$, for $\mathsf{e}_i, \ldots, \mathsf{e}_{i'} \in A_\mathsf{E}$. Consider the following cases: (Sub-Case $\mathsf{e}_0 \cup \mathsf{f}_0 \in A_\cup$.) Then, in either of the next two possibilities, $[\mathsf{U}, \mathsf{E}_m]$ is defined as an $\mathbb{M}$-action:

$$[\mathsf{U}, \mathsf{E}_m] = \begin{cases} [\mathsf{U}, \mathsf{e}_{i'} \otimes \cdots \otimes \mathsf{e}_i] & \text{if } \pi \neq \pi_\varnothing(\mathsf{e}_0 \cup \mathsf{f}_0, \ldots) \\ [\mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0] & \text{otherwise} \end{cases}$$

(Sub-Case $\mathsf{e}_0 \cup \mathsf{f}_0 \in A_\uplus$.) Let $\pi = \pi_\varnothing(\ldots, \mathsf{e}_0 \cup \mathsf{f}_0, \mathsf{f}_j, \ldots, \mathsf{e}'_0 \cup \mathsf{f}'_0, \ldots)$, where, $\mathsf{e}'_0 \cup \mathsf{f}'_0$ is the next action not in A_E after $\mathsf{e}_0 \cup \mathsf{f}_0$. Then, in either of the next two possibilities, we obtain the same $\mathbb{M}$-action, that is

$$[\mathsf{U}, \mathsf{E}_m] = \begin{cases} [\mathsf{U}, (\mathsf{e}_0 \otimes \mathsf{e}_{i'} \otimes \cdots \otimes \mathsf{e}_i) \cup (\mathsf{f}_0 \otimes \mathsf{f}_{j'} \otimes \cdots \otimes \mathsf{f}_j)] \\ \qquad\qquad \text{if } \pi = \pi_\varnothing(\ldots, \mathsf{e}_0 \cup \mathsf{f}_0, \mathsf{f}_j, \ldots, \mathsf{f}_{j'}; \ldots, \mathsf{e}'_0 \cup \mathsf{f}'_0, \ldots) \\ [\mathsf{U}, (\mathsf{e}_0 \otimes \mathsf{e}_{i'} \otimes \cdots \otimes \mathsf{e}_i) \cup (\mathsf{f}_0 \otimes \mathsf{f}_{j'} \otimes \cdots \otimes \mathsf{f}_j)] \\ \qquad\qquad \text{if } \pi = \pi_\varnothing(\ldots, \mathsf{e}_0 \cup \mathsf{f}_0, \mathsf{f}_j, \ldots, \mathsf{f}_{j'}; \mathsf{e}'_0 \cup \mathsf{f}'_0, \ldots) \end{cases}$$

Note that by Definition 7.4.4, these two are the only possibilities. First, observe that such action $\mathsf{f}_{j'}$ must exist, since π is a finite sequence of refinements on the

empty plan $\pi_\varnothing$. Second, if π is of the form $\pi_\varnothing(\ldots, \mathsf{e}_{j'}, \mathsf{e}'_0 \cup \mathsf{f}'_0)$, the refinement with $\mathsf{e}'_0 \cup \mathsf{f}'_0$ would not be valid, since it would apply to a partial action E_m.

(Sub-Case $\mathsf{e}_0 \cup \mathsf{f}_0 \in A_{\underline{\cup}}$.) The proof is analogous to the previous sub-case.

(Sub-Case no action $\mathsf{e}_0 \cup \mathsf{f}_0$ exists in π.) Then $\pi = [\mathsf{U}, \mathsf{E}_m] = [\mathsf{U}, \mathsf{e}_{i'} \otimes \cdots \otimes [\mathsf{U}, \mathsf{e}_i]$, and we are done.

(Ind. case $k+1 \to k$) Assume that $\pi_{k+1} = [\mathsf{U}, \mathsf{E}_{k+1}] \ldots [\mathsf{U}, \mathsf{E}_m]$ is a sequence of $\mathbb{M}$-actions. And let $\pi = \pi_\varnothing(\ldots, \mathsf{e}^\star, \ldots)$ be such that $\mathsf{e}^\star$ is the first action in π not used in the definition of π_{k+1}. By replacing φ_G by $\varphi_{\mathsf{goals}(\pi_{k+1})}$, we can repeat the same reasoning as in the base case from $\mathsf{e}^\star$ onwards to generate some $\mathbb{M}$-action E_k. Again, if all the actions from $\mathsf{e}^\star$ onwards are in A_E, we also conclude $\pi = [\mathsf{U}, \mathsf{E}_k], \ldots, [\mathsf{U}, \mathsf{E}_m]$.

Finally, since the actions in π are finite, there exists a finite number of actions e^* in this plan which are not in A_E, say m'. Then this process ends up with a finite sequence of $\mathbb{M}$-actions $[\mathsf{U}, \mathsf{E}_1] \ldots [\mathsf{U}, \mathsf{E}_m]$ where $m' \leq m \leq m'+1$. Thus, π is a $\mathbb{M}$-sequence.

Note first, that now we can fix a particular $\mathrm{LCC}_{\cup\otimes n}$ logic, given by the action model U used by $\mathbb{M}$, and a bound n defined as the maximum length r of the $\mathbb{M}$-actions E_k in the output π. (Where the length of $\mathsf{E}_k = \mathsf{e} \cup \mathsf{f}$ is the maximum of lengths of e and f.) For simplicity, though, we simply keep the previous notation $[\mathsf{U}, \cdot]$, rather than using $[\mathsf{U}^{\leq r}, \cdot]$.

Let us re-enumerate the $\mathbb{M}$-sequence given by the output π now as $[\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_1]$. Let also π_k denote the plan corresponding to the $\mathbb{M}$-sequence $[\mathsf{U}, \mathsf{E}_k] \ldots [\mathsf{U}, \mathsf{E}_1]$. The proof that $[\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_1]$ is a solution for $\mathbb{M}$ is by induction on the length k of initial sub-sequences $[\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}]$. Let π_k denote the $\mathbb{M}$-sequence $[\mathsf{U}, \mathsf{E}_k] \ldots [\mathsf{U}, \mathsf{E}_1]$. First we show the claim

$$\models \varphi_{\mathsf{goals}(\pi_{k+1})} \to ([\mathsf{U}, \mathsf{E}_{k+1}]\varphi_{\mathsf{goals}(\pi_k)} \wedge \langle \mathsf{U}, \mathsf{E}_{k+1}\rangle \top)$$

(Sub-Case $\mathsf{E}_{k+1} \in A_\mathsf{E}{}^{<\omega}$.) Let $\mathsf{E}_{k+1} = \mathsf{e}_i \otimes \cdots \otimes \mathsf{e}_{i'}$. Observe that in this case, the construction of the $\mathbb{M}$-action E_{k+1} proceeds as in the case of deterministic planning (Def. 5.2.2) with a deterministic planning domain $\mathbb{M}_{k+1} = (\varphi_{\mathsf{goals}(\pi_{k+1})}, A_\mathsf{E}{}^{\leq r}, \varphi_{\mathsf{goals}(\pi_k)})$ (except that now we explicitly composite actions in $A = A_\mathsf{E}{}^{\leq r}$). Hence, the proof of the claim can be obtained by the soundness result Theorem 5.3.1, together with the fact that $[\mathsf{U}, \mathsf{e}_i] \ldots [\mathsf{U}, \mathsf{e}_{i'}]$ can equivalently be replaced by $[\mathsf{U}, \mathsf{e}_i \otimes \cdots \otimes \mathsf{e}_{i'}]$, that is by $[\mathsf{U}, \mathsf{E}_{k+1}]$; and similarly for $\langle \mathsf{U}, \mathsf{e}_i\rangle \ldots \langle \mathsf{U}, \mathsf{e}_{i'}\rangle$ and $\langle \mathsf{U}, \mathsf{E}_{k+1}\rangle$. It can be observed, then, that the fact that E_{k+1} is a deterministic solution for $\mathbb{M}_{k+1}$ is equivalent to the desired claim above.

(Sub-Case $\mathsf{E}_{k+1} \in A_\cup$.) Let $\mathsf{E}_{k+1} = \mathsf{e}_0 \cup \mathsf{f}_0$. Then we simply have that

$$\begin{array}{lcll} \varphi_{\mathsf{goals}(\pi_{k+1})} & = & t([\mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0]\varphi_{\mathsf{goals}(\pi_k)} \wedge \langle \mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0\rangle \top) & \text{(Def. 7.4.2)} \\ & \equiv & [\mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0]\varphi_{\mathsf{goals}(\pi_k)} \wedge \langle \mathsf{U}, \mathsf{e}_0 \cup \mathsf{f}_0\rangle \top & \text{(correctness of } t\text{)} \end{array}$$

(Sub-Case $\mathsf{E}_k = (\mathsf{e}_0 \otimes \mathsf{e}_{i'} \otimes \cdots \otimes \mathsf{e}_i) \uplus (\mathsf{f}_0 \otimes \mathsf{f}_{j'} \otimes \cdots \otimes \mathsf{f}_j)$.) The reasoning is similar to the previous case, now using the definition of $\varphi_{\mathsf{goals}(\pi_\varnothing(\ldots,\mathsf{f}_{j'};))}$.

(Sub-Case $\mathsf{E}_k = (\mathsf{e}_0 \otimes \mathsf{e}_{i'} \otimes \cdots \otimes \mathsf{e}_i) \underline{\cup} (\mathsf{f}_0 \otimes \mathsf{f}_{j'} \otimes \cdots \otimes \mathsf{f}_j)$.) The proof is again similar. The only difference is that in addition now we have that $\models \varphi_{\mathsf{goals}(\pi_{k+1})} \to [a](\mathsf{pre}(\mathsf{e}_0) \wedge \neg\mathsf{pre}(\mathsf{f}_0)) \vee [a](\neg\mathsf{pre}(\mathsf{e}_0) \wedge \mathsf{pre}(\mathsf{f}_0))$.

Finally, we can show the soundness of the output plan π. The proof is by induction on the claims that

$$\models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}] \varphi_{\mathsf{goals}(\pi_k)} \quad \text{and} \quad \models \varphi_T \to \langle \mathsf{U}, \mathsf{E}_m \rangle \ldots \langle \mathsf{U}, \mathsf{E}_{k+1} \rangle \top$$

This will suffice, since for the particular case of 0, we have, $\varphi_{\mathsf{goals}(\pi_0)} = \varphi_{\mathsf{goals}(\pi_\varnothing)} = \varphi_G$, so the claim shows that π is a solution for $\mathbb{M}$.

(Base Case m) We show that $\models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \varphi_{\mathsf{goals}(\pi_{m-1})}$ and $\models \varphi_T \to \langle \mathsf{U}, \mathsf{E}_m \rangle \top$. Since the Terminating Condition is satisfied by the output $\pi = \pi_m$ we have $\models \varphi_T \to \varphi_{\mathsf{goals}(\pi_m)}$. By the above proof, we also have $\models \varphi_{\mathsf{goals}(\pi_m)} \to [\mathsf{U}, \mathsf{E}_m] \varphi_{\mathsf{goals}(\pi_{m-1})}$ and $\models \varphi_{\mathsf{goals}(\pi_m)} \to \langle \mathsf{U}, \mathsf{E}_m \rangle \top$. From these and the previous fact, the above claims immediately follow.

(Ind. Case $k+1 \Rightarrow k$) Assume (Ind. Hyp.) that the claim holds for the initial fragment $[\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}]$; that is, $\models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}] \varphi_{\mathsf{goals}(\pi_k)}$ and $\models \varphi_T \to \langle \mathsf{U}, \mathsf{E}_m \rangle \ldots \langle \mathsf{U}, \mathsf{E}_{k+1} \rangle \top$. We proceed to show the two claims:

$\models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}] \varphi_{\mathsf{goals}(\pi_k)}$	(Ind. Hyp.)
$\models \varphi_{\mathsf{goals}(\pi_k)} \to [\mathsf{U}, \mathsf{E}_k] \varphi_{\mathsf{goals}(\pi_{k-1})}$	(shown above)
$\models [\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}] \left(\varphi_{\mathsf{goals}(\pi_k)} \to [\mathsf{U}, \mathsf{E}_k] \varphi_{\mathsf{goals}(\pi_{k-1})}\right)$	(Nec.)
$\models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}] [\mathsf{U}, \mathsf{E}_k] \varphi_{\mathsf{goals}(\pi_{k-1})}$	(K)

$\models \varphi_T \to \langle \mathsf{U}, \mathsf{E}_m \rangle \ldots \langle \mathsf{U}, \mathsf{E}_{k+1} \rangle \top$	(Ind. Hyp.)
$\models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}] \varphi_{\mathsf{goals}(\pi_k)}$	(Ind. Hyp.)
$\models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \ldots [\mathsf{U}, \mathsf{E}_{k+1}] \langle \mathsf{U}, \mathsf{E}_k \rangle \top$	(shown above)
$\models \varphi_T \to \langle \mathsf{U}, \mathsf{E}_m \rangle \ldots \langle \mathsf{U}, \mathsf{E}_{k+1} \rangle \langle \mathsf{U}, \mathsf{E}_k \rangle \top$	(1st and 3rd claims))

□

Theorem 7.4.8 (Completeness)**.** *For a given planning domain $\mathbb{M}$, if some $\mathbb{M}$-sequence exists that is a solution to $\mathbb{M}$, then the BFS method from Algorithm 5 terminates (with a solution).*

Proof. Without loss of generality, in this proof we will assume that $\mathsf{pre}(\mathsf{e})$ is consistent for each $\mathsf{e} \in \mathsf{E}$ and that $\mathsf{pre}(\mathsf{e} \otimes \cdots \otimes \mathsf{e}')$ is consistent for each existing $\mathsf{e} \otimes \cdots \otimes \mathsf{e}' \in A_{\mathsf{E}}{}^{<\omega}$.

The search space is a finitely-branching tree (with root $\pi_\varnothing$ and child nodes in A), and thus the algorithm terminates provided some plan π exists that satisfies

the Terminating Condition. Assuming a solution exists, let us check that some plan π exists that is an $\mathbb{M}$-sequence satisfying the Terminating Condition.

Let $[\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_1]$ be a $\mathbb{M}$-sequence which is a solution for $\mathbb{M}$. Without loss of generality, we can assume that this $\mathbb{M}$-sequence has minimal total length, where the length of $(\mathsf{e}_{i_1}, \dots, \mathsf{e}_{i_r})$ is r, and the length of $\mathsf{E}_j = (\mathsf{e}_0 \otimes \mathsf{e}_{i_1} \otimes \dots \otimes \mathsf{e}_{i_r}) \cup (\mathsf{f}_0 \otimes \mathsf{e}_{j_1} \otimes \dots \otimes \mathsf{e}_{j'_r})$ is $1 + r + r'$. Moreover, we can assume that the solution $[\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_1]$ is a $\mathbb{M}$-sequence satisfying $\mathsf{E}_j \in {A_\mathsf{E}}^{<\omega}$ iff $j = 1$ or $\mathsf{E}_{j+1} \in A_\cup$. To see this can be assumed, we can first equivalently replace all the $[\mathsf{U}, \mathsf{e}] \dots [\mathsf{U}, \mathsf{f}]$ modalities with $\mathsf{e}, \dots, \mathsf{f} \in A_\mathsf{E}$ as a single modality $[\mathsf{U}, \mathsf{e} \otimes \dots \otimes \mathsf{f}]$. Moreover, suppose that $[\mathsf{U}, \mathsf{E}_{j+1}][\mathsf{U}, \mathsf{E}_j]$ is of the form

$$\mathsf{E}_j \in {A_\mathsf{E}}^{<\omega} \quad \text{and} \quad \mathsf{E}_{j+1} = (\mathsf{e}_0 \otimes \mathsf{e}_i \otimes \dots \otimes \mathsf{e}_{i'}) \cup (\mathsf{f}_0 \otimes \mathsf{f}_j \otimes \dots \otimes \mathsf{f}_{j'})$$

with $\cup \in \{\uplus, \bigcup\}$. Then, by iteratively replacing all these pairs $[\mathsf{U}, \mathsf{E}_{j+1}][\mathsf{U}, \mathsf{E}_j]$ with the corresponding modality

$$[\mathsf{U}, (\mathsf{e}_0 \otimes \mathsf{e}_i \otimes \dots \otimes \mathsf{e}_{i'} \otimes \mathsf{E}_k) \cup (\mathsf{f}_0 \otimes \mathsf{f}_j \otimes \dots \otimes \mathsf{f}_{j'} \otimes \mathsf{E}_k)]$$

the resulting expression is an $\mathbb{M}$-sequence of the desired form.

We show that $[\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_1]$ is generated by the BFS algorithm and satisfies the Terminating Condition –if no other plan is generated first, in which case we would also be done. As usual we will refer by π_k to the $\mathbb{M}$-sequence (or plan) $[\mathsf{U}, \mathsf{E}_k] \dots [\mathsf{U}, \mathsf{E}_1]$. In addition, without loss of generality, we also assume that the solution $[\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_1]$ is minimal in the sense that no actions from A can be deleted from it while preserving success and executability.

The proof of the present Theorem is by induction on the construction of the solution. Since, though, the Base Case for $[\mathsf{U}, \mathsf{E}_1]$ and the Ind. Case $[\mathsf{U}, \mathsf{E}_k] \dots [\mathsf{U}, \mathsf{E}_1] \Rightarrow [\mathsf{U}, \mathsf{E}_{k+1}][\mathsf{U}, \mathsf{E}_k] \dots [\mathsf{U}, \mathsf{E}_1]$ are similar, we only prove the former. For the inductive case, one can simply replace φ_G by $\varphi_{\mathsf{goals}(\pi_k)}$.

(Case $\mathsf{E}_1 \in {A_\mathsf{E}}^{<\omega}$.) The proof is similar to that of the Completeness Theorem 5.3.1 for deterministic planning in the domain $(\varphi_{\mathsf{goals}(\pi_1)}, A_\mathsf{E}, \varphi_G)$, except that the plan $\pi_\varnothing(\mathsf{e}_{i'}, \dots, \mathsf{e}_i)$ now defines an $\mathbb{M}$-action $[\mathsf{U}, \mathsf{E}_1] = [\mathsf{U}, \mathsf{e}_i \otimes \dots \otimes \mathsf{e}_{i'}]$ rather than as a sequence $[\mathsf{U}, \mathsf{e}_i] \dots [\mathsf{U}, \mathsf{e}_{i'}]$. Theorem 5.3.1 also shows that no action $\mathsf{e} \in \{\mathsf{e}_i, \dots, \mathsf{e}_{i'}\}$ makes the corresponding plan $\pi_\varnothing(\dots, \mathsf{e})$ a leaf.

(Case $\mathsf{E}_1 \in A_\cup$) It is obvious that $[\mathsf{U}, \mathsf{E}_1]$ is constructible since $\mathsf{E}_1 \in A_\cup$. Let $\mathsf{E}_1 = \mathsf{e}_0 \cup \mathsf{f}_1$. To see that $\pi_\varnothing(\mathsf{e}_0 \cup \mathsf{f}_0)$ is not a leaf, we reason as follows. By definition of planning domain φ_T is consistent, so let M, w be a model for φ_T. Then,

$M, w \models \varphi_T$

$M, w \models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_2] \left([\mathsf{U}, \mathsf{E}_1]\varphi_G \wedge \langle \mathsf{U}, \mathsf{E}_1 \rangle \top \right)$ (by def. of solution)

$M, w \models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_2] t \left([\mathsf{U}, \mathsf{E}_1]\varphi_G \wedge \langle \mathsf{U}, \mathsf{E}_1 \rangle \top \right)$ (correctness of t)

$M, w \models \varphi_T \to [\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_2] \varphi_{\mathsf{goals}(\pi_\varnothing(\mathsf{e}_0 \cup \mathsf{f}_0))}$ (Def. 7.4.2)

$((M \circ \mathsf{U}) \dots \circ \mathsf{U}), ((w, \overrightarrow{\mathsf{e}_m}), \dots, \overrightarrow{\mathsf{e}_2}) \models \varphi_{\mathsf{goals}(\pi_\varnothing(\mathsf{e}_0 \cup \mathsf{f}_0))}$
for any $\overrightarrow{\mathsf{e}_m} \in \mathsf{E}_m, \dots, \overrightarrow{\mathsf{e}_2} \in \mathsf{E}_2$

so we conclude that $\varphi_{\mathsf{goals}(\pi_\varnothing(\mathsf{e}_0 \cup \mathsf{f}_0))}$ is consistent.

(Case E_1 contains a $\uplus$ or $\underline{\cup}$ modality.) Let $\mathsf{E}_1 = (\mathsf{e}_0 \otimes \mathsf{e}_i \otimes \dots \otimes \mathsf{e}_{i'}) \cup (\mathsf{f}_0 \otimes \mathsf{e}_j \otimes \dots \otimes \mathsf{e}_{j'})$, with $\cup \in \{\uplus, \underline{\cup}\}$. A simple proof by induction with Definitions 7.4.4 and 7.4.3 shows that $\pi_\varnothing(\mathsf{e}_{i'}, \dots, \mathsf{e}_i, \mathsf{e}_0 \cup \mathsf{f}_0, \mathsf{e}_{j'}, \dots, \mathsf{e}_j;)$ is constructible, since the deterministic refinements are not leafs. Using this, the proof that E_1 is not a leaf, for the case $\cup = \uplus$, is analogous to the previous case with $\cup$. Finally, the same holds for the case where $\cup = \underline{\cup}$: the proof is analogous to that for $\uplus$, together with the immediate fact that if $\varphi_{\mathsf{goals}(\pi_\varnothing(\dots, \mathsf{e}_0 \uplus \mathsf{f}_0, \dots, \mathsf{e};)}$ is consistent, then so is the formula

$$\left([a](\mathsf{pre}(\mathsf{e}_0) \wedge \neg \mathsf{pre}(\mathsf{f}_0)) \vee [a](\mathsf{pre}(\mathsf{f}_0) \wedge \neg \mathsf{pre}(\mathsf{e}_0)) \right) \wedge \varphi_{\mathsf{goals}(\pi_\varnothing(\dots, \mathsf{e}_0 \uplus \mathsf{f}_0, \dots, \mathsf{e};)}$$

But the latter is precisely identical to $\varphi_{\mathsf{goals}(\pi_\varnothing(\dots, \mathsf{e}_0 \uplus \mathsf{f}_0, \dots, \mathsf{e};)}$, so we are done. This, the plan $\pi_\varnothing(\dots, \mathsf{e}_0 \cup \mathsf{f}_0, \dots, \mathsf{e};)$ corresponds to the $\mathbb{M}$-action $[\mathsf{U}, \mathsf{E}_1]$.

As we said, the proof of the inductive case is analogous. Thus, we conclude that a plan π is constructible. The proof concludes by observing that after constructing $\pi_m = [\mathsf{U}, \mathsf{E}_m] \dots [\mathsf{U}, \mathsf{E}_1]$, the Terminating Condition is satisfied. The claim $\models \varphi_T \to \varphi_{\mathsf{goals}(\pi_m)}$ follows from inspection of the definition in each case: $\mathsf{E}_k \in A_\mathsf{E}$; $\mathsf{E}_k \in A_\cup$; and finally E_k of the form $[\mathsf{U}, (\mathsf{e}_0 \otimes \dots \otimes \mathsf{e}') \cup (\mathsf{f}_0 \otimes \dots \otimes \mathsf{f}')]$, with $\cup \in \{\uplus, \underline{\cup}\}$. □

Example 7.4.9 (Coin toss announcement)**.** For a problem involving the use of $\mathbb{M}$-actions containing $\underline{\cup}$, consider the following: a coin was tossed into the agent a's hand, where it remains unobserved. The agent a wants to know whether it landed heads or tails, and a also wants that the (gullible) agent b believes it landed heads. Let the planning domain $\mathbb{M} = (\varphi_T, A, \varphi_G)$ defined by

$$\varphi_T = \top, \quad A = \{\, h!_b^{[a]} \underline{\cup} h\dagger_b^{[a]},\ \mathsf{feel}_h \uplus \mathsf{feel}_{\neg h} \,\}, \quad \varphi_G = ([a]h \vee [a]\neg h) \wedge [b]h$$

.

Then $[\mathsf{U}, \mathsf{feel}_h \uplus \mathsf{feel}_{\neg h}][\mathsf{U}, h!_b^{[a]} \underline{\cup} h\dagger_b^{[a]}]$ is a solution to $\mathbb{M}$. Note that in this example,

$$\varphi_{\mathsf{goals}(\pi_\varnothing(h!_b^{[a]} \underline{\cup} h\dagger_b^{[a]}))} = ([a]h \vee [a]\neg h) \wedge ([a]([a]h \wedge \langle a \rangle h) \vee [a]([a]\neg h \wedge \langle a \rangle \neg h))$$

Since these goals are satisfied by the sensing action, which has no preconditions, the resulting plan has as open goals $\top$, and hence it is executable and it satisfies the Terminating Condition.

In some planning scenarios, one can only build plans which *might* lead to a goal state, also called weak solutions. In this cases, the study of weak non-deterministic planning is also of interest; that is, the study of systematic search algorithms for this kind of plans. The previous definitions and results easily adapt to the case of plan search for weak solutions.

Definition 7.4.10 (Weak solution)**.** We say that an $\mathbb{M}$-sequence $[\mathsf{U}, \mathsf{E}_1], \ldots, [\mathsf{U}, \mathsf{E}_r]$ is a *weak solution* for a planning domain $\mathbb{M} = (\varphi_T, A, \varphi_G)$ iff we replace the (*success*) condition in Def7.3.6 by

$$\models \varphi_T \to \langle \mathsf{U}, \mathsf{E}_1 \rangle \ldots \langle \mathsf{U}, \mathsf{E}_r \rangle \varphi_G \quad (\textit{poss. success})$$

The definitions of plan refinement Def. 7.4.2–7.4.3 are analogously defined except that now we replace in the definition of $\varphi_{\mathsf{goals}(\pi(\cdot))}$ the condition

$$t\left([\mathsf{U}, \mathsf{E}_k]\varphi_{k+1} \wedge \ldots\right) \qquad \text{by} \qquad t\left(\langle \mathsf{U}, \mathsf{E}_k \rangle \varphi_{k+1} \wedge \ldots\right)$$

The results of soundness and completeness are analogously proved for the same Breadth First Search based Algorithm 5 (except that now this algorithm is based in the new definitions of plan refinement). In other words, if the algorithm terminates given some non-deterministic planning domain $\mathbb{M}$, it outputs a weak solution for $\mathbb{M}$. And if a weak solution for $\mathbb{M}$ exists, then Algorithm 5 terminates with such a solution.

Theorem 7.4.11 (Soundness and Completeness for Weak Planning)**.** *The Algorithm 5 is sound and complete for weak non-deterministic planning in* $\mathrm{LCC}_{\cup\otimes n}$.

7.5 Conclusions and Related Work

We presented backward planning algorithms for a planner-executioner agent with LCC-reasoning abilities. These algorithm enables the agent to find deterministic or (non-deterministic) strong plans in multi-agent scenarios. The logics considered here are dynamic epistemic logics with ontic or physical actions. This permits that plans contain communicative actions, sensing or the usual fact-changing actions. Hopefully, the proposed methods might be used for practical reasoning in communicative agents, and in particular provide a logical foundation for the modeling (and computing) of agreements among motivated agents.

In the literature on planning, a study of strong non-deterministic plan search can be found in [41] for classical planning. This work also deals with non-deterministic iteration *, or cyclic planning. In relation to this, as commented in 6.6, there are computational issues preventing similar results in the corresponding extensions of LCC with non-deterministic iteration. For a study of strong planning under partial observability, see [23]. The main difference between this and the approaches based on dynamic epistemic logic, is that in the former, classical planning is extended with new components, e.g. an observation function is added to the structure of a planning domain. In the present approach,

in contrast, one rather extends the language and logic underlying classical planning. As a consequence, the traditional structure of planning domains, i.e. as triples *(initial state, actions, goals)*, is preserved in the resulting planning systems. See Section B.2 for a more detailed comparison between standard planning approaches and the present work based on the LCC dynamic epistemic logics.

In the area of logic, a backward approach to planning is considered in [49], for the case of public actions (i.e. announcements and assignments). More in line with the present approach, one can find in the literature several works by Bolander et al. In [28] an approach to forward planning is considered based on the DEL logics from [142]. This approach semantically defines the space of states and the available actions. Thus, the planning states are multi-pointed epistemic models $M, \{w, \ldots, w'\}$ that obtain after update with arbitrary plans π (analogous to the semantics of the present planning states $[\![\varphi_{\mathsf{goals}(\pi)}]\!]$ in backward planning); similarly, the planning actions are pointed action models U, E_d, rather than the corresponding modalities $[\mathsf{U}, \mathsf{E}_d]$ in a language $\mathcal{L}^{\cup}_{\mathsf{U}}$. This paper studies the (semi-)decidability of single- and multi-agent planning based on Breadth First Search for forward plan search. The main differences between the planning system from [28] and the present approach are: the use of semantic or proof-theoretic tools; a different base epistemic logic with: single-agent and common knowledge modalities for groups, $[a]$ and $[B^*]$, or the E·PDL-modalities of LCC. In [9], the authors study single-agent conditional planning in the DEL logic [142]. This work combines the epistemic modalities $[a], [B^*]$ and the dynamic modalities U, e from [142], with composition $\otimes$ and a conditional action constructor

$$\textit{if } [a]\varphi \textit{ then do } \pi \textit{ else do } \pi'$$

where a denotes the planner agent a. This proposal for non-deterministic planning system is based on forward search on AND/OR trees, in contrast to the present framework. Besides this difference and the present multi-agent approach, none of the two planning systems seems to extend the other one in terms of expressivity; on the one hand, our notion of conditional plan is limited to a sequence of conditional actions, while in [9] conditional actions can be nested: the above π and π' can again contain conditional constructors. On the other hand, in this chapter we did not assume that the non-deterministic actions are distinguishable immediately before execution, and for example the actions defined by $\cup$ and $\uplus$ need not be so.

Conclusions and Open Problems

In the two Parts of this thesis, we have proposed methods to solve problems of practical reasoning using techniques from the area of planning. The motivation for the proposed methods was to combine the expressivity and tools from the two areas of logic and planning into logic-based planning systems. This resulted in flexible planning algorithms for the construction of plans that can be expressed in the corresponding logics.

The main contributions of this dissertation are soundness and completeness theorems of these planning algorithms, as well as the study of the logical properties of the t-DeLP logic programming system, and soundness and completeness of the LCC logics extended with composition and choice. The contributions of this thesis can be summarized as follows:

- the procedure for t-DeLP warrant in the classes of simple and mutex programs satisfies the properties of consistency and closure called Rationality Postulates.
- Breadth First Search for backward planning is sound and complete for the class of planning domains expressible in some simple or mutex t-DeLP logical program.
- a dialogue-based Breadth First Search planning algorithm for multiple collaborative planner agents is sound and complete w.r.t centralized planning.
- Breadth First Search is sound and complete for deterministic backward planning in the planning system defined by any LCC logic.
- the logics extending LCC with composition and choice are sound and complete w.r.t. a natural extension of their axioms and semantics.
- Breadth First Search is sound and complete for strong non-deterministic (backward) planning in any LCC logic extended with composition and choice.

As a comparative summary of the concepts and methods used in the two Parts of this dissertation, see the next table.

	in Part I	in Part II
select a decidable **logical system**:	argumentation-based logic programming (without logical symbols for action)	a dynamic logic (i.e. with action modalities)
Model (or state, in logic)	a truth-assighment on Var	a static model M (i.e. an epistemic model)
State **description** (init. state and goals)	a set of literals	a formula in the static fragment
State (in planning)	a logical program, or the partial truth-assignment induced by it	a set of models for the open goal φ, i.e. $[\![\varphi]\!]^M$
Determ. **Actions** of an agent (or attributed to it)	definable pairs *(preconditions, effects)* each a set of literals	atomic action modalities
Available actions	all actions	some subset of actions
Planning domains	(log. prog., actions, goals)	(init., actions, goal)
Dynamic **transitions**	as in state-transition systems	product update
Action **Update**	syntactic replacement, if preconditions are true	semantically defined, if preconditions are true
Plans	given by the planning system	programs constructible from atomic actions
Search space	(planning) states	(planning) states
Search methods	BFS, DFS, etc.	BFS
Refinement steps	action-supported proofs (or arguments)	actions
Plan **Threats**	logical inconsistencies or counter-arguments	(none in linear planners for monotonic logics)
Threat maneuvers	manually adapting those of the planning system	
Open goals after refinement	as in planning: unsolved goals plus new actions' preconditions	syntactic reduction into the static fragment of the language
Terminating condition for the search algorithm	initial state includes open goals; no threats exist	initial state implies open goals

Next we summarize each Part, and list some interesting question from each chapter that remain as open problems.

Part I In this part, we proposed a logic programming system t-DeLP for temporal defeasible reasoning, and studied its argumentation-theoretical properties. We also proposed a planning system based on this logic and studied different planning algorithms for solving the corresponding problems of practi-

cal reasoning. From this first part of the thesis, we would like to emphasize the advantages in terms of knowledge representation and, more specifically, of the *dynamics of* knowledge representation. Hopefully, this was shown throughout the Examples from Chapter 1.3.3 for t-DeLP logic programming. In these examples, the expansion of a knowledge base with more precise information (rules with new atoms) is straightforward and can be done without modifying the previous representations. A bit more surprising are the examples in Section 2.2 about the introduction of actions. In these examples, a simple action description was expanded with more refined actions, in a *monotonic* way. In contrast, the same maneuver in classical planning (based on monotonic logic) would require a totally new action description from scratch. In general, the dynamics of knowledge representation in monotonic logics typically exhibit a non-monotonic character: when more precise information must be introduced into a knowledge base, this knowledge base needs first a full revision.

This advantage is most important in the study of multi-agent argumentation [135], and decentralized planning systems as in Chapter 3. The reason is that the initial knowledge bases of these agents might be the result of heterogenous experiences from previous interactions with the environment. That is, these agents might have learnt by induction different lessons from the different context they have previously experienced (defeasible rules). This allows that the unique condition in decentralized argumentation or planning is that of a global consistency on the strict information. For the case of mutex programs, this consistency requirement affects both the hard facts of agents' experience (strict facts), and the conceptual constraints (mutex rules). This makes the consistency requirement to be reasonably weak for the tasks of multi-agent argumentation and planning in the class of t-DeLP mutex programs.

Chapter 1 A first question left unanswered in this thesis is the computational complexity of the warrant procedure for t-DeLP. A conjecture would be that the complexity is the same than in the DeLP case; according to [39], the question whether a literal is warranted in a program is in some class NP^C, where C is some parametrized class given by the particular defeat criteria.

From the point of view of the language, t-DeLP is still far from the expressivity of temporal logic programming systems [1], [20]. On the one hand, one might like to extend the t-DeLP framework in order to accept arbitrary queries, that is, whether some general strict or defeasible rule is warranted given a logical program. (For strict queries, at least, the application of the techniques from [20] should be straightforward.) On the other hand, it is hard to guess how difficult might it be to expand the t-DeLP language of Chapter 1 with temporal modalities like *until*, although this would also be an interesting question to pursue. Finally, *past* temporal operators might be introduced to address post-diction questions, or (evidence-based temporal) reasoning from the presently observable effects to its presumable causes in the past. In the same line, the t-DeLP state transition system from Chapter 2 could be extended with such *past* operators for counterfactual reasoning about the past (with alternative plans that were not executed).

More immediate open questions, also related to Chapter 1, include the extension of the consistency and closure postulates to more general classes of t-DeLP logic programs, e.g. programs containing arbitrary strict rules without delay. It can be conjectured that the *closure under transposition* for strict rules might suffice for this. (see Section C.2).

Finally, extensions or revisions of the current defeat relation(s) for t-DeLP are also conceivable. For example, a preference for *more precise* temporal information might be considered in addition to (or in place of) the current preference criteria based on a preference for *more facts* and *less persistence*.

Chapter 2 As mentioned in this chapter, severe restrictions were imposed on the temporal actions in t-DeLP planning, since most of the responsibility was left to the set of defeasible rules. Along this line, it would be interesting to consider more expressive representations of temporal actions, like those usual in temporal planners: actions with preconditions at different times, with arbitrary durations, with preconditions that must hold during all the execution, and so on. Another line of research, closer to least-commitment policies, would be to replace linear plans with more flexible plan representations like those of GraphPlan and POP, in line with [62] for DeLP.

Other interesting topics for future research are comparative studies in the computational complexity of forward and backward t-DeLP plan search. Finally, the study of heuristics might provide some clues to reasonably good implementations of the algorithms proposed. Due to the defeasible character of t-DeLP, though, traditional heuristics for planning do not seem to work.

Chapter 3 In this last chapter of Part I, some open problems have been left for future work. First, it would be interesting to perform experimental tests for a comparison between the centralized and distributed algorithms from Chapters 2 and 3. These would reveal some parameters and values for which one or the other approach is better suited in terms of computational costs. Among these parameters, one might consider the number of dialoguing agents, or the size of the planning domain(s). The use of more flexible planners for t-DeLP could also be extended to the present multi-agent planner approach, in line with [111]. Finally, in the literature on multi-agent planning, the merging of plans by different agents (or groups) into a single consistent joint plan has also become a focus of interest. This might be done as well in t-DeLP planning by adapting the proposed dialogues to this task.

Part II In the second part of this thesis, we focus on the extension of dynamic epistemic logic with composition and choice, and studied search algorithms for the resulting planning systems. This permitted to address the issues of partial observability and non-determinism. The main conclusion of this second part is an increase in the kind of epistemic scenarios that can be addressed by planning algorithms. This is not only due to the existence of (nested) epistemic modalities, but also to the existence of a rich class of epistemic actions, including communicative and sensing actions. Needless to say, the advantages the proposed planning systems are inherited from the base logics LCC [139] (and

$\mathrm{LCC}_{\otimes\cup n}$), and are due to the expressive power of these logics.

In comparison, in view of the technical issues existing in the logics of intentions (see Section 5.4) it seems altogether more practical to drop the intention modalities and instead adopt methods based on plan search. The present approach might be applied as well to other dynamic logics (without intentions), to obtain practical reasoning systems.

Chapters 4 and 6 The field of dynamic epistemic logic is currently a very active area, and an ever-growing number of extensions of basic DEL logics can be found in the literature. For example, the introduction of *concurrency* of actions, considered in [143], would permit more general planning systems, where agents execute actions at the same time. This is interesting provided that time is implicitly represented in dynamic epistemic logic. Similarly, the introduction of dynamic operators for *belief revision* has been much studied, e.g. [146, 147, 137]. An extension of the present planning techniques into LCC logics with belief revision operators would increase the flexibility of the communicative abilities of the present LCC planners.

Chapters 5 and 7 Following the comparison with [9] in Section 7.5, a natural question is whether one can combine the concept of conditional plan in this work with the present notion of non-deterministic actions (as well as with a backward approach to multi-agent planning). Some of the technical issues that would arise were discussed during the introduction of non-deterministic modalities in Chapters 6 and 7.

Another question related to [28] is the study of the decidability and complexity of the proposed planning algorithms. The results in these parts show that multi-agent (strong) planning is at least semi-decidable, so the results for backward planning are not worse than those from a forward approach. The decidability of single-agent planning in the present case is an open problem.

At the level of applications, the study of heuristic criteria for particular languages of LCC or $\mathrm{LCC}_{\cup\otimes n}$ logics seem also of interest. This topic seems to demand a brand new approach for DEL logics w.r.t. well-known heuristic criteria from classical planning. The reason would be the delicate epistemic issues involved in logical proofs (for the construction of correct plans) within dynamic epistemic logics.

An extension of the language of goals would also be of interest in order to distinguish between implication goals $\varphi \to \psi$ (*the agent goal is that* $\varphi \to \psi$ *is true*) and conditional goals (e.g. (whenever the agent believes that φ is true, then her set of goals is expanded with ψ). An action making φ false or ψ true would solve the implication goal; but only the latter would solve the conditional goal.

More general problems related to the results in these chapters are described below, together with possible lines of research for those problems.

Parts I and II Let us conclude this section with further important issues and questions that remain open for the logics and the planning systems considered in either Part I or Part II of this dissertation.

A first issue, already commented in the above paragraphs is the computational complexity of the different algorithms used in the proposed logics and planning systems, as well as empirical comparisons between the different algorithms considered.

A second important topic in the literature on planning, which we did not mention, is that of *optimal planning* or the search of solutions which are optimal in terms of some notion of cost, see Appendix A. Often, planning actions are associated with some cost, expressing the cost of a single application of an action. This cost function is considered additive, so the cost of a (deterministic) plan is the sum of the costs of its actions. Natural notions of cost exist for physical or epistemic actions, in terms of energy consumption or brevity of communicative actions. In relation to optimal planning, the Breadth First Search planning methods proposed, easily extend to Best First Search, e.g. uniform search. The same can be said about the proofs of soundness and completeness of these planning methods, which can be extended without much effort towards optimal planning. Thus, the cost of a plan in the framework of LCC (deterministic) planning is the sum of its actions. For t-DeLP planning the cost of a plan step $\mathcal{A}$ in t-DeLP planning is the sum of costs of the actions supporting $\mathcal{A}$ not already in the plan. These extensions might find applications in several fields of research oriented to optimization that share interests with t-DeLP (optimization in industry or robotics applications based on temporal reasoning) or with LCC logics (some areas of linguistics and cognitive science).

As we mentioned, the methods proposed in Parts I and II might as well apply to other temporal or dynamic logics with similar properties to those of t-DeLP and, respectively, LCC. That is, logics without actions, or logics with axiomatically characterized actions. Certainly, there still exists a full spectrum of intermediate cases lying between these two extreme positions considered in this thesis. These intermediate cases include those logics that contain symbols for actions, but characterize them in the knowledge base, rather than through the axioms of the logic. Among them, we can list several logics with dynamic modalities (like PDL or different logics for actions) as well as logics which represent (the actual execution of) an action simply as a new atomic variable. It remains open what stance is most promising in order to define planning systems for these logics.

Another important open problem along the proposed lines of research is whether the present results on planning extend to *strategic reasoning*. On the one hand, the literature on planning suggests that appropriate search methods in OR-graphs can solve planning problems in most of the existing planning domains, and these are also the methods used in this thesis. On the other hand, certain classes of game-theoretic problems can be solved with the existing methods for search in AND/OR-graphs. Whether these methods can be shown to be sound or even complete for the planning systems inspired by t-DeLP or LCC must be left as future work.

Finally, there remains the difficult problem of higher-order practical reasoning (e.g. *I believe that you intend that I believe that you intend that* ψ.) Surely

this kind of reasoning was one of the motivations for the logics of intentions; certainly, this also became the source of the problems associated with its high expressive power. It is hard to imagine how the combined logic and planning methods, for example, could be embedded again for some logic for second- or higher-order practical reasoning. A more traditional approach, in the line of the logics of intention, might consist in combining dynamic epistemic logics with some modal logics of preference, in order to capture mechanism for plan selection within the logic.

In summary, this thesis leaves a considerable number important questions as open problems. Some of these can be pursued with the help of the techniques proposed in this thesis, while others might demand new approaches and points of view.

Part III

Appendixes

This Appendix contains an overview of some of the areas related to this dissertation, with an emphasis on those parts used in the previous Chapters. These areas are the following:

- **Automated Search** is the study of algorithms (and their properties) that explore arbitrary graphs in a class in order to find some designated nodes. A graph-based representation has proved successful as a general-purpose problem solving technique. The idea is to view problem solving as an incremental process of solution construction (or exploration) that takes place in the corresponding graph representation. In this sense, different types of graphs seem to capture natural classes of problems, whenever these are appropriately represented. Many problems in planning, logic, decision and game theory can be expressed within this representation, and hence can be solved by the corresponding algorithms.

- **Planning** studies the representation and generation of plans for some class of practical problems, given by propositional representation of states, goals and the components of actions (preconditions, effects). The class of problems addressed by the algorithms proposed for a planning system mainly depends on the expressivity of the latter. We present the traditional paradigms in planning as an introduction to this area, and also to motivate the proposed extensions of standard planners with richer underlying logical frameworks from Chapter 1 and Chapter 4. Comparison with traditional planners, corresponding to each part of the thesis, are also proposed.

- **Argumentation** is a recent area for the study of inference methods based on argumentative processes (i.e. how arguments attack each other). It aims to formalize the deliberative aspects of dialogues that aim to resolve the truth-value of a claim in dispute. Following its historical development, this area is divided into two parts: an abstract study of the semantics based on the attack relation; followed by some logical foundations for these elements, that make the internal logical structure of arguments explicit, and naturally induce the relation of attack between such structured arguments.

Appendix A

Search

Automated search [115] (or simply Search) is the general study of problem solving techniques for problems admitting graph-theoretic representations. A graph can represent a space of partial solutions, which are built according to some procedure in an incremental way. Two partial solutions (or nodes) are related by an arc if one of them obtains from the other by refinement with some further condition imposed upon the simpler node. A solution is a partial solution meeting all the requirements specified by the problem.

Different types of search problems can be identified, for example

- search: find a solution
- optimal search: find a solution which is at least as good as any other existing solution,
- near-optimal search: find a solution that approximates the value of the best existing solution

In this chapter, we review some search algorithms for the first or second kind of problems. In the first case, one is particularly interested in techniques that are sound and complete for a given class of graphs. These mean, respectively, that the search algorithm only terminates with solutions, and that a solution is found if some solution exists. This problems addressed in Chapters 2, 3, 5 and 7 are of this kind.

In optimal search, the goal is moreover to find some solution that minimizes some notion of solution cost. In simple cases, this notion of solution cost is additive, i.e. the cost of a solution is the sum of the costs of each condition added to it during its construction. It is not difficult to adapt certain search algorithms like Breadth First Search for tasks of optimal search e.g. Best First Search.

The different techniques studied in the literature perform better or worse on different classes of graphs, depending on their size and branching factor. The literature restricts its attention to locally finite graphs (countable, finitely

branching). Though some of the studied search methods can be used as well in (recursively) enumerable graphs or trees, the class of locally finite graphs does suffice for the present purposes: the logics from Parts I and II only admit finitely-many actions.

In addition to these results, the literature has devoted considerable efforts on the question of heuristic search. These are functions for the estimation of the extra cost needed yet to reach a solution from the partial solution under consideration. This estimation can be defined for example in terms of the difference between the current partial solution and the requirements for solutions in the present problem. The topic of heuristic search falls out of the scope of this dissertation, though. Considerable efforts are devoted in the area of planning to finding heuristic functions for given planning domains. For a full-length treatment of these topics, the reader is referred to the textbook [115].

A.1 Problems represented in graphs and trees.

For many worldly problems, *understanding a problem* seems to imply having a definite idea of all its candidate solutions (or how to generate them), and also knowing how a solution would look like.

Example A.1.1 (TSP)**.** Consider, for example, the Travel Salesman Problem (TSP) problem of finding the shortest route through a set C of n cities. Anyone understanding this formulation must understand as well what an arbitrary route in this set of cities is, e.g. a circular list $c_1 \to \ldots \to c_n \to c_1$, here represented as a sequence $(c_1, \ldots, c_n)$. Also, a notion of cost can be naturally assigned to the possible routes, e.g. in terms of length (or expected energy consumption). Let us assume in addition that the solver knows (how to compute) the length of any route. E.g the solver might know the distances $d(i, j)$ between each pair of cities i, j, so the length of route $(c_1, \ldots, c_n)$ is $d(c_1, c_2) + \cdots + d(c_n, c_1)$. To test whether $(c_1, \ldots, c_n)$ is an optimal solution, one can compare it to the rest of solutions in terms of such length.

In general, if (1) all solutions (whichever they are) are known to be in a given *well-defined set*, and moreover (2) a *solution test* exists for arbitrary elements of this set, then solving such a problem would just reduce to look (and test) into these elements according to, say, some well-ordering of this set. If the (finite) set of nodes is big enough, though, it will be unpractical to make this set explicit.

In practice, then, it is better to have a *constructive representation* for the set of solutions, so each solution candidates can be obtained as the result of a series of decisions made during the construction process. These decisions are imposed one by one until no other decisions need to be added (a solution was just built) or can be added.

Example A.1.2 (TSP; cont'd)**.** Following the description of the TSP problem in Example A.1.1, any route $(c_1, \ldots, c_n)$ obtains from (consistently) selecting pairs $(i, j) \in (C \times C)$ (with $i \neq j$) expressing the condition *after visiting i, visit*

j. Note that (i, j) is inconsistent with pairs (i, j'), with (j, i), and (i', j), for any $j' \neq j$ and $i' \neq i$.

Indeed, in TSP any route $(c_1, \ldots, c_n)$ obtains as a (maximally) consistent subset of decisions, i.e. $\{(c_1, c_2), \ldots, (c_{n-1}, c_n)\}$. Let us remark that, for TSP and many other problems, the order in which these decisions are taken does not matter: the same partial solution is ultimately built from the same elements in any order of refinement. This suggests a graph representation, where a node can be reached from different paths: an expansion of $\{(c_1, c_2)\}$ with condition (c_2, c_3) results in the same node that an expansion of $\{(c_2, c_3)\}$ with (c_1, c_2). Namely, the partial solution $\{(c_1, c_2), (c_2, c_3)\}$. In general, though, this order-independence cannot be assumed, in which case the problem is represented as a tree. See Figure A.2 below.

Definition A.1.3 (Graph; Path)**.** A *graph* is a pair (N, R), where N is a set and R a relation on N; i.e. $R \subseteq N \times N$. These two elements are called nodes N and arcs R. A *path* is any sequence of nodes $(\nu_0, \ldots, \nu_k)$ such that $R(n_i, n_{i+1})$ for each $0 \leq i < k$. If R is asymmetric (N, R) is called a *directed graph*.

The graphs studied in search are assumed to have a unique start node ν_0, which informally represents the (unique) empty partial solution.

Definition A.1.4 (Successor and parent nodes)**.** For a given graph (N, R), we define the *set of successors* of some $\nu \in N$, denoted $R(\nu)$ or $R(\nu, \cdot)$, as follows

$$R(\nu) = \{\nu' \in N \mid R(\nu, \nu')\}$$

The *set of parent nodes* of some $n \in N$, represented as $R^{-1}(\nu)$, is analogously defined as $R^{-1}(\nu) = \{\nu' \in N \mid R(\nu', \nu)$.

Definition A.1.5 (Tree)**.** A *tree* is a graph (N, R) such that all nodes, except for the start node, have a single parent node, i.e. $|R^{-1}(\nu)| = 1$.

Assume that the above requirements (1)-(2) are met for a given problem, and moreover that all of the solutions of this problem can be represented as the result of a series of decisions on their construction. In this case, the problem can be captured as a graph or tree (N, R). Its set of nodes $N = \{\nu, \ldots\}$ will contain the empty node ν_0, partial solutions and solutions.

These partial solutions are given by a series of choices on some finite set of *decisions* $\mathsf{A} = \{\mathsf{e}_1, \ldots, \mathsf{e}_n\}$.[1] For example, in the TSP problem, the set A is the set of pairs of cities (c_i, c_j). The set of partial solutions is either the product $\mathsf{A}^{\leq \omega}$ or some bounded set $\mathsf{A}^{\leq n}$ or simply some Cartesian product A^n. The TSP problem is in the latter class, since a solution route must consist of n pairs (c_i, c_j) where $n = |C|$ is the number of cities in the problem.

Thus, the arc relation R in the graph representation of a problem (N, R) can be obtained by means of a *refinement* operator $\oplus$ which maps a partial solution

[1] This notation A and e_i reflects the notation for planning in Chapters 5 and 7. For planning problems, the set of decisions $\mathsf{A} = \{\mathsf{e}, \ldots\}$ corresponds to the set of actions e available to the agent.

ν and a decision e into a more constrained partial solution $\nu \oplus \mathsf{e}$. The set of arcs in the graph can thus be represented as well as follows

$$R(\nu, \nu') \qquad \Leftrightarrow \qquad \nu' = \nu \oplus \mathsf{e}, \quad \text{for some decision } \mathsf{e} \in \mathsf{A}$$

The graphs defined above are called OR-graph in the literature, in order to distinguish them from AND/OR graphs (see below). An OR-graph models the structure of partial solutions ν with OR-nodes, since the refinement of such a partial solution ν takes place by choosing either of the decisions e_1 or $\ldots$ or e_n in the set A.[2]

The representations based on OR-graphs are useful for finding solutions to problems that depend upon these decisions (controlled by our solver agent), and possibly other "decisions" from some unintentional external cause, like nature. With more generality, a scenario might include an external agent (or agents) with an influence upon the outcome of this construction, e.g. modifying it with their own decisions. If these decisions are known as well by our solver, the resulting structure of partial solutions give rise to an AND/OR graph. An AND/OR graph is an hypergraph with two types of nodes: OR-nodes (as before, representing our solver decisions) and AND-nodes (representing the decisions from an external agent).

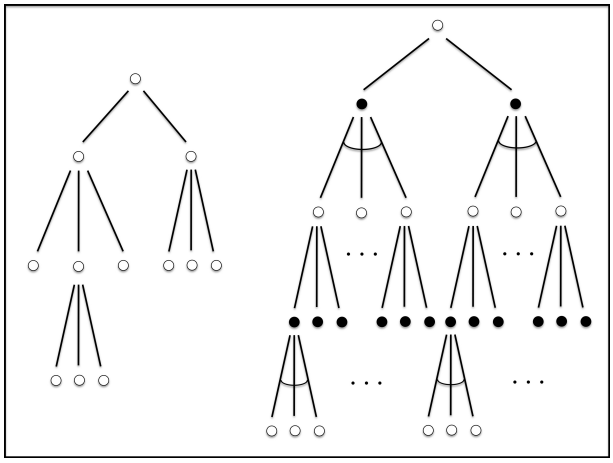


Figure A.1: (Left) An OR-graph; (Right) An AND/OR-graph, or hypergraph, where a related set of solid nodes denote an external agent's choice.

Automated search is the study of algorithms that explore (or unfold) graphs of either form. The differences between search algorithms consist in the different control strategies that are used to decide which parts of the graph are to be

[2] All of the planning systems considered in the previous chapters are represented by OR-graphs: in Part I, the set of decisions is the set of plan steps (argument steps or threat resolution moves), though only those that are applicable to the present node ν will give rise to refined nodes. In Part II, the set of decisions is the set of available actions, also denoted A.

generated and explored first (or next). The desirable theoretical properties of these algorithms are listed next:

- Soundness: if the search algorithm terminates with a partial solution ν, this is a solution
- Completeness: if a solution exists, the algorithm outputs a solution
- Admissibility: the output of the algorithm is optimal among the other solutions

In the simple example of the TSP problem with $n = |C|$ cities, the soundness and completeness properties are quite trivial, since solutions have a specific form, i.e. as elements of $\mathsf{A}^{|C|}$. (This is not in general the case, though.) The TSP problem is interesting when the solver tries to find an optimal solution.

In the following, we describe some of the best known techniques for tentative search. *Tentative* here denotes those search methods that maintain a list of rival partial candidates yet to be explored (open nodes), rather than considering at each step a single partial candidate, to be increasingly refined up to a unique terminal node. Search methods of the latter kind (e.g. hill-climbing) are called irrevocable search, because one always commits to explore a single partial candidate, and forgets about its previous refinement history. In irrevocable search one can only start again from scratch, if a dead end (a non-solution terminal node), is ultimately found.

When search is restricted to combining actions into a plan, as in the chapters of this dissertation, one can simplify the search space into a tree (rather than a graph, see Figure A.2). That is, we can assume that the order in which we add the conditions from A is essential to the resulting node. This is the case for deterministic plans as in Chapter 5, where the order in which actions are added is the reverse of their execution order according to the plan. This assumption is natural under planning with implicit time, since the actions in this kind of plans are not in general order-independent, e.g. consider the actions *turn on the light* and *climb down the ladder*.

See Figure A.2, for an illustration of the graph- and tree-representations of a problem. The corresponding sets of N contain, resp., sets and sequences of elements in A. The set of solutions is denoted $\mathtt{Sol}$.

$$\begin{aligned} N = &\quad \{\, \emptyset,\ \mathsf{a},\ \mathsf{b},\ \mathsf{c},\ \mathsf{a}\oplus\mathsf{b},\ \ldots,\ \mathsf{c}\oplus\mathsf{b}\,\} \\ \mathtt{Sol} = &\quad \{\, \mathsf{a}\oplus\mathsf{b},\ \mathsf{b}\oplus\mathsf{a},\ \mathsf{a}\oplus\mathsf{c},\ \mathsf{c}\oplus\mathsf{a}\} \end{aligned}$$

In the case of a graph, where nodes are sets, we have $|N| = 7$ and $|\mathtt{Sol}| = 2$. In the tree representation of the same problem, where nodes are sequences, we have $|N| = 10$ and $|\mathtt{Sol}| = 4$.

A distinction already mentioned above is that between uninformed and informed search algorithms. In the step-wise, incremental construction of possible solutions, *uninformed* search methods define a control strategy on how to prioritize the set of refinements $\{\nu \oplus \mathsf{e}_1, \ldots, \nu \oplus \mathsf{e}_n\}$ of the current node ν, w.r.t. the previously generated nodes ν' still pending of evaluation. *Informed* search

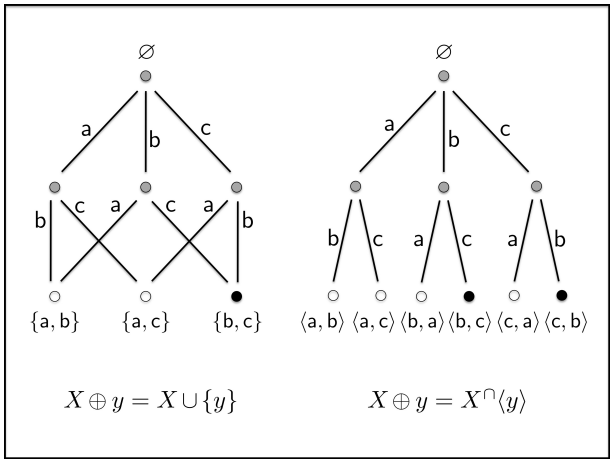


Figure A.2: Two similar search problems, in graph (left) and tree form (right) given by, resp., commutative and non-commutative refinement operators $\boxtimes$.

defines, in addition, a priority ordering within the set of open nodes, including both the old and the newly generated nodes in $\{\nu \oplus \mathsf{e}_1, \ldots, \nu \oplus \mathsf{e}_n\}$. To this end, an heuristic method can make use of the information upon the current node ν, e.g. its cost $\mathsf{cost}(\nu)$, as well as using a heuristic function $f(\nu)$ that estimates the additional cost of obtaining a solution node ν^* from the current ν. Heuristic methods require that these estimations are optimistic, that is, that the real cost of a solution ν^* extending ν is at least the estimated cost for ν, namely $\mathsf{cost}(\nu) + f(\nu) \geq \mathsf{cost}(\nu^*)$. The heuristic information $\mathsf{cost}(\nu) + f(\nu)$ describes how much promising the current node ν is. This information is used to prioritize search by exploring the more promising nodes first.

In the (tsp) optimization problem, the distance between two cities $d(c_i, c_j)$ in a partial route ν containing (c_i, c_j) is an example of factual information. The sum of the distances of the conditions $\mathsf{e} \in \nu$ in this node, $f(\nu) = \Sigma_{(c_i,c_j)\in\nu} d(c_i, c_j)$ indeed gives a lower bound for the cost of any solution extending ν. An heuristic estimation $h(\nu)$ for this node, for instance, can be given by a random completion ν^* of this route ν. The estimation in terms of ν^* can give a (possibly misleading) estimation of the cost of arbitrarily extending the node ν to a solution.

A.2 Uninformed search in OR-graphs (trees): BFS, DFS.

One of the well-known exhaustive uninformed search algorithms is that of Breadth First Search (BFS), see Figure A.3(Right). Define the *depth* of a partial solution ν is given by the number of refinement steps $\mathsf{e} \in \mathsf{A}$ needed to define ν. The idea of BFS is perform a (potentially complete) exploration of the search space by considering partial solutions with increasing depth. Breadth-first is a

FIFO (first in first out) based search algorithm, since latest (more refined) nodes are explored later. The BFS method for a search spaces in tree form is described next in Algorithm 7.

Data: A tree (N, R); a root note $\nu_0 \in N$
a refinement oper. $\oplus$; a test function $\texttt{test}(\cdot) : N \to \{\top, \bot\}$.
Result: ν; or fail
initialization: $\nu = \nu_0$ and $\mathsf{Open} = \langle \nu_0 \rangle$;
while $\texttt{test}(\nu) = \bot$ **do**
  delete ν from Open;
  set $\mathsf{Open} = \mathsf{Open}^{\frown}\langle \nu \oplus \mathsf{e} \mid \mathsf{e} \in \mathsf{A} \rangle$;
  set $\nu =$ the first element of Open;
end

Algorithm 6: Breadth First Search in some search space in tree form.

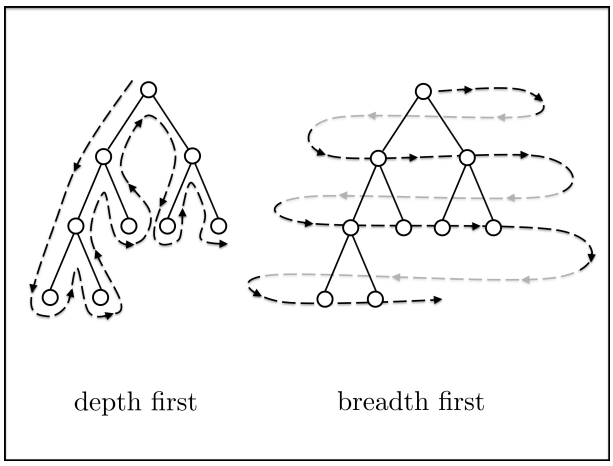


Figure A.3: An illustration of the ordering of node exploration in finite graphs: (Left) Depth First Search; (Right) Breadth First Search.

It is instructive to compare BFS with its LIFO counterpart, called Depth-First Search (DFS); see Figure A.3(Left). In the DFS method, the newly generated refinements $\nu \oplus \mathsf{e}$ of a node ν are explored before the previously generated nodes still pending of evaluation. DFS can be defined from BFS by prefixing the set Open with the latest generated nodes, rather than suffixing it with latest nodes. More formally, DFS is defined by replacing in the above BFS method, the condition

$$\mathsf{set\ Open} = \mathsf{Open}^{\frown}\langle \nu \oplus \mathsf{e} \mid \mathsf{e} \in \mathsf{A} \rangle$$
$$\text{by the condition}$$
$$\mathsf{set\ Open} = \langle \nu \oplus \mathsf{e} \mid \mathsf{e} \in \mathsf{A} \rangle^{\frown}\mathsf{Open}$$

After we expand a node into its immediate refinements, one of those is further selected, whenever possible, and otherwise it proceeds with some unexplored node that was generated earlier. DFS is based on ideas similar to those after irrevocable search methods, despite DFS is revocable itself.

DFS is not complete for countable trees nor, in particular, for those locally finite trees having infinite paths. The reason is that it may keep expanding further and further a node in a forever useless way. This makes a depth-bound on depth-first search is usually added to plain DFS in order to regain completeness. See below for a summary of the basic properties of these algorithms.

A.3 Informed search in OR-graphs: Best First Search

The above uninformed search algorithms BFS, DFS, etc. have counterparts in informed search, when some heuristic estimation of nodes is introduced. A heuristic estimation $\mathsf{cost}(\nu) + f(\nu)$ can be used to sort the set of remaining or open nodes, in an increasing way w.r.t. this cost estimation. Thus

Data: A tree (N, R); a root note $\nu_0 \in N$
a refinement oper. $\oplus$; a test function $\mathtt{test}(\cdot) : N \to \{\top, \bot\}$
a cost function $\mathsf{cost} : N \to \mathbb{R}$; a heuristic function $f : N \to \mathbb{R}$
Result: ν; or fail
initialization: $\nu = \nu_0$ and $\mathsf{Open} = \langle \nu_0 \rangle$;
while $\mathtt{test}(\nu) = \bot$ **do**
  delete ν from Open;
  set $\mathsf{Open} = \langle \nu' \in \mathsf{Open} \cup \{\nu \oplus \mathsf{e}\}_{\mathsf{e} \in \mathsf{A}} \rangle$;
    (where ν' occurs before ν'' if $\mathsf{cost}(\nu') + f(\nu') \leq \mathsf{cost}(\nu'') + f(\nu'')$);
  set $\nu =$ the first element of Open;
end

Algorithm 7: Best First Search in some search space in tree form.

Note that an optimistic heuristic function is always definable, namely as the trivial function $f(\nu) = 0$. The resulting method is called *uniform search.*

Theorem A.3.1. *[115] BFS is complete in locally-finite trees. DFS is complete in finite trees. Best-First Search is admissible under any optimistic heuristic function f.*

Appendix B

Planning

Planning is the task of generating plans for given practical problems. The latter, called **planning domains**, are at least described by a triple

initial state s_0	describing (partial knowledge upon) the current state upon which the plan executioner is to act
avail. actions A,	from which the planner can freely choose to build the plan, and
goal states S_g	describing a set of states (or the conditions that make them desirable)

Automated planning is the study of general algorithms for a class of planning domains. The latter is given by some propositional-based language, used to describe states and actions, e.g. as a pair of sets of propositions (preconditions, effects). An action is executable in some state if the preconditions hold, and in case this action is executable, the resulting, the effects of the action hold in the resulting state.

A **plan** is some some structure (e.g. a sequence) of actions, prescribing what is to be done at each step during its execution. A plan must specify the next step in an unambiguous way, though possibly depending on observations made during this execution. For the plan to be a **solution** for a given planning domain, the execution of the plan must lead to some goal state in S_g, when this plan execution takes place in the initial state s_0.

Structure of the chapter.

In this chapter, we review some standard planning systems from the literature in Section B.1 (classical planning) and in Section B.2. The latter in particular briefly describes some extensions of classical planning, closely related to the planning systems from Part I and Part II. These are temporal planning systems and non-deterministic partially-observable planning.

In Section B.2, a comparison is made between some of these planning systems and those proposed of in Chapters 2, 5 and 7.

B.1 Classical Planning.

Classical planning [66], offers a simple conceptual model to formalize the previous basic elements. It is based on a discrete model for events, called (restricted) state transition systems.

Definition B.1.1 (State transition system)**.** A *restricted state transition system* is a tuple $\Sigma = \langle S, A, R\rangle$, where

- S is a finite (or recursively enumerable) set of states,
- A is a finite set of actions, and
- $R : S \times A \to S$ is a computable transition function. This transition function associates is partial, and for any $(s, \mathsf{e}) \in S \times A$, either $R(s, \mathsf{e})$ is undefined or an element $R(s, \mathsf{e}) \in S$.

A planning problem, called planning domain, can be defined by designated initial and goal states in given state transition systems.

Definition B.1.2 (Classical planning domain.)**.** A *classical planning domain*is a triple (Σ, s_0, S_g), where

- Σ is a restricted state transition system
- $s_0 \in S$ is the *initial state*
- $S_g \subseteq S$ is the set of *goal states*

In this simple state-based approach, plans are finite sequences of actions. Plans of this kind are also called linear or totally ordered plans.

Definition B.1.3 (Plan. Solution.)**.** A *plan* in a given classical planning problem (Σ, s_0, S_g) is a finite sequence of actions $\pi = \langle \mathsf{e}_1, \mathsf{e}_2, \ldots, \mathsf{e}_m\rangle$. A *solution* to (Σ, s_0, S_g) is some plan such that

$$R(R(\ldots R(R(s_0, \mathsf{e}_1), \mathsf{e}_2), \ldots, \mathsf{e}_{n-1}), \mathsf{e}_n) \in S_g.$$

Since the set of actions is finite, any classical planning problem (Σ, s_0, S_g) is at least semi-decidable: a planner algorithm can explore (using a breadth-first method) the space of states reachable from s_0, while testing whether any of these is an element of S_g. In case a solution exists, some solution will eventually be found.

In the set-theoretic representation of classical planning, it is assumed a (finite) propositional representation of states by means of a set of atomic variables $\mathsf{Var} = \{p, q, \ldots\}$, i.e. the corresponding set of literals $\mathsf{Lit} = \mathsf{Var} \cup \{\neg p \mid p \in \mathsf{Var}\}$. A state can be represented either as a mapping from Var to $\{0, 1\}$ (a model), a subset of Var, or a consistent and complete collection of literals $\ell \in \{p, \neg p\}$ for each $p \in \mathsf{Var}$.

Actions in planning also follow this propositional encoding of states. Thus, the map between states $R(\cdot, \mathsf{e}) : S \to S$ that encodes action e, is induced by a pair of preconditions and effects. In the following, for a given literal $\ell \in \{p, \neg p\}$ and set of literals X, we define

$$\neg\ell = \begin{cases} \neg p & \text{if } \ell = p \\ p & \text{if } \ell = \neg p \end{cases}$$
$$\neg X = \{\neg\ell \mid \ell \in X\}$$

Definition B.1.4 (Action)**.** An action is a pair $\mathsf{e} = (\mathsf{pre}(\mathsf{e}), \mathsf{post}(\mathsf{e}))$ of consistent sets of literals, where

- $\mathsf{pre}(\mathsf{e}) \subseteq \mathsf{Lit}$ denotes the *preconditions* for action e to be executable in state s:
$$R(s, \mathsf{e}) \in S \qquad \text{iff} \qquad \mathsf{pre}(\mathsf{e}) \subseteq s$$
- $\mathsf{post}(\mathsf{e}) \subseteq \mathsf{Lit}$ denotes the *effects* of actions e upon a state s (where e is executable in s):
$$\ell \in R(s, \mathsf{e}) \qquad \text{iff} \qquad \ell \in \mathsf{post}(\mathsf{e}) \text{ or } (\ell, \neg\ell \notin \mathsf{post}(\mathsf{e}) \text{ and } \ell \in s)$$

Thus, in planning, a state-transition function is induced by the so-called update or progression function.

Definition B.1.5 (Update function)**.** Assume a set of propositional states $S = \{s : \mathsf{Var} \to \{0, 1\}\}$ and a set A of actions $\mathsf{e} = (\mathsf{pre}(\mathsf{e}), \mathsf{post}(\mathsf{e}))$ are given. The induced *update function*, resp. defined for actions a and plans $\pi = \langle \mathsf{e}_1, \ldots, \mathsf{e}_m \rangle$) is

$$R(s, \mathsf{e}) = \begin{cases} (s \setminus \neg\mathsf{post}(\mathsf{e})) \cup \mathsf{post}(\mathsf{e}) & \text{if } \mathsf{pre}(\mathsf{e}) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases}$$
$$R(s, \langle \mathsf{e}_1, \ldots, \mathsf{e}_m \rangle) = \begin{cases} s & \text{if } m = 0 \text{ (i.e. } \pi = \varnothing) \\ R(R(s, \mathsf{e}_1), \langle \mathsf{e}_2, \ldots, \mathsf{e}_m \rangle) & \text{if } m > 0 \text{ and } \mathsf{pre}(\mathsf{e}_1) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases}$$

Any such triple $\Sigma = (S, A, R)$ of states, actions and update function is a state-transition system.

Finally, for backward planning, one computes the regression of the open goals (current goals) by an action. This step provides the new goals to be achieved. That is, it gives the minimal conditions (propositions) required for an arbitrary state to make e executable in a way that leads to the former open goals. In this approach, a planner only needs to consider *relevant* actions that can contribute to the ultimate goals, expressed by S_g.

Definition B.1.6 (Relevance. Regression.)**.** Given a planning domain (Σ, s_0, S_g), a goal state $s \in S$ (initially $s \in S_g$) and an action $\mathsf{e} \in A$, we say e is *relevant* for s iff $s \cap \mathsf{post}(\mathsf{e}) \neq \varnothing$ and $s \cap \neg\mathsf{post}(\mathsf{e}) = \varnothing$. In case e is relevant for s, we define the regression of goal s by e as follows

$$R^{-1}(s, \mathsf{e}) = (s \setminus \mathsf{post}(\mathsf{e})) \cup \mathsf{pre}(\mathsf{e})$$

A search method like BFS or DFS, see Chapter A can be applied to forward or backward state-based planning using, resp., the above notions of *executable action* e *in a state* s and *relevant action* e *for a state* s. Classical planning has also been encoded into satisfiability problems (SAT), by encoding each possible action execution as a formula. A planning problem is captured by a formula expressing *initial state* $\wedge$ *all possible actions* $\wedge$ *goal*. A SAT-solver aims to find a model for it, encoding a solution plan. This plan consists in those action executions whose formulas are assigned true in the model.

Different languages for planning systems have been considered. First, STRIPS listed only the positive literals in states (based on the *closed world assumption* that unlisted literals are false); the action description language ADL expanded the language of states with negative facts and a built-in equality, as well as quantifiers and disjunction for goals, and finally conditional effects for actions. These languages are included in standard contemporary languages called PDDL.

B.2 Beyond Classical Planning: lifting assumptions.

In summary, classical planning [66] addresses planning domains that are definable in state-transition systems Σ satisfying the following conditions:

A1 finite Σ: the set of states S is finite, and so is the set of actions A. This assumption is enforced by taking the set of atoms Var to be finite itself.

A2 fully observable Σ: the planner always knows exactly which is the actual state. i.e. the truth-value of each variable p in the current state is always known.

A3 deterministic Σ: each action in the state-transition system is defined by a function $R(\cdot, a)$ assigning each state s a single state.

A4 static Σ: no other events take place besides those actions planned and executed by the agent.

A5 restricted goals: goals are conditions. defining a set of ultimately desired states; in particular, the way the plan is to reach some such state is of no concern.

A6 sequential plans: plans are sequences of actions.

A7 implicit time: the representation of state transitions does not make any quantitative notion of time explicit; i.e., the timing and duration of actions is not considered. Time is qualitatively partitioned by the actions according to the sequential plan.

A8 offline planning: planning algorithms solve a planning domain, and then the resulting plan is executed.

The study of planning systems that drop some of the assumptions A1-A8 has been an important aim among the contemporary literature on planning.

Dropping A6 has received much attention. Besides the state-based planning approach for linear planners, that order the plan steps (as sequences) during the construction of the plan, more flexible representations of plans have been proposed in planning. For example, GraphPlan [27] groups plan steps that do not interfere with each other (and can be ordered arbitrarily); Partial Order Planning (POP) [116] manages a list of ordering constraints between actions, only expanded when deemed necessary. These approaches, which search for groups of equivalent linear plans, are inspired by least-commitment principles. This has been done in parallel to the study of heuristics for planning. Heuristic search has mainly been studied in two main directions: search on relaxed planning problems, that abstract from the preconditions of actions or their negative effects; or by assuming a logical independence among sub-goals.

Dropping some of these assumptions A1-A8 is also the aim in the present logic-based approach. For example, the planning systems in PartI based on t-DeLP defeasible temporal logic programming in Part I drop the assumptions A2, A4 and A7. The planning systems in Part II based on dynamic epistemic logic LCC drop the A2 and A3 assumptions. In the literature, lifting subsets of the assumptions A1-A8 has been accomplished through different planning systems. In the following section, we describe non-deterministic planning in partially-observable domains.

Temporal planning.

An in-depth study of the relation of the t-DeLP planning system with temporal planners falls out of the scope of this chapter, since it involves technical details related to binding constraints (in planning) and unification in temporal logic programming. In Chapters 2 and 3, we skipped these technical issues, and presented the planning system with temporal constants and instantiated rules and actions. As in much of the literature on planning, most temporal planners build flexible plans based on least-commitment principles. (This is useful when the executing agents are homogeneous w.r.t. their abilities, and can replace each other for most tasks.)

Let us point out that the basic elements, e.g. in a chronicle-based approach, can be understood in t-DeLP as general defeasible rules and schemas of temporal literals. We briefly comment upon some relations between a t-DeLP and a chronicle-based approach to temporal planning.

Definition B.2.1 (Temporal assertion)**.** A *temporal assertion* on a state variable $p(x)$ is either of the following

- an event $p@t : (v_1, v_2)$, specifying an instantaneous change at t, from $p(v_1)$ to $p(v_2)$
- a persistence condition $p@[t_1, t_2) : v$, specifying the persistence of $p(v)$ during the interval $[t_1, t_2)$.

This approach consists in extending classical planning with state variables, e.g. $p(x)$ (or more generally, $p(x_1, \ldots, x_n)$) which are functions of time. This time-oriented approach, in contrast with state-oriented views, does not require axioms such as *an object is at a single place at one time*, because of the use of state variables.

Definition B.2.2 (Chronicle)**.** A *chronicle* for a given set of variables $\{p, \ldots, q\}$ is a pair $\Phi = (\mathcal{F}, \mathcal{C})$ where $\mathcal{F}$ is a set of temporal assertions about $\{p, \ldots, q\}$ and $\mathcal{C}$ is a set of object constraints and temporal constraints (on the corresponding object or temporal variables).

Chronicles are used to specify temporal actions, without distinguishing between preconditions and effects. For example, the event $p@t : (v, v')$ describes, in terms of t-DeLP, both a precondition $\langle p(v), t-1\rangle$ and an effect $\langle p(v'), t\rangle$. These temporal assertions are also used to specify the initial state and goals. A temporal operator can be applied to a chronicle, and so on.

Comparison with t-DeLP.

The previous elements can be translated into the t-DeLP planning from Chapter 2. For example, temporal assertions can be expressed as temporal facts and rules

- an event $p@t : (v_1, v_2)$ can be expressed as a set $\{\langle p(v_1), t-1\rangle, \langle p(v_2), t\rangle\}$,
- a persistence condition $p@[t_1, t_2) : v$ can be seen a set of general persistence rules $\{\delta_{p(v)}(t)\}_{t_1 \leq t < t_2 - 2}$,

where v, v_1, v_2 are object variables and t, t_1, t_2 are temporal variable. The use of state-variables can be expressed with the corresponding set of mutex constraints.

Our conjecture is that the t-DeLP planning system can encode an arbitrary temporal planning domain (in the chronicle-based approach) as a sub-class of the t-DeLP planning domains based on mutex programs. With more detail, as a planning domain $\mathbb{M} = ((\Pi, \Delta), A, G)$ where $\Pi = \Pi_{\mathsf{f}} \cup \Pi_{\mathbf{M}}$,

- Π_{f} is a set of temporal literals encoding the initial state's events (i.e. temp. assertions)
- $\Pi_{\mathbf{M}}$ is a set of mutes rules encoding the state-variable representation
- Δ_{p} encodes the persistence conditions in the plan construction
- $\Delta \setminus \Delta_{\mathsf{p}}$ encodes the (direct) effects of temporal operators Φ_{e}, e.g. in $p@t : (v_1, v_2)$ as an indirect effect rule $\langle p(v_2), t\rangle \multimap \langle \mu_{\mathsf{e}}, t-1\rangle$
- A encodes the instantiations of temporal operators

Note that the use of Δ rules to encode the direct effects can be generalized to any instantiated temporal operator of an arbitrary finite duration, say from t to $t+n$. This can easily be encoded with a tuple of temporal actions $\mathsf{e} = \mathsf{e}^1; \ldots; \mathsf{e}^n$ defined by: (1) $\mathsf{pre}(\mathsf{e}^1) = \mathsf{pre}(\mathsf{e})$, (2) $\mathsf{post}(\mathsf{e}^n) = \mathsf{post}(\mathsf{e})$, and (3) for the

remaining preconditions and effects, new set of variables μ_{e^k} are introduced into the language, each exclusive to the corresponding pair $\mathsf{e}^k, \mathsf{e}^{k+1}$ of consecutive actions: $\mathsf{post}(\mathsf{e}^k) = \langle \mu_{\mathsf{e}^k}, t + k \rangle = \mathsf{pre}(\mathsf{e}^{k+1})$. We also conjecture that other temporal aspects in the rich descriptions of temporal operators can be, with more considerable effort, encoded in the t-DeLP planning framework as well.

Planning in non-deterministic partially-observable domains

Dropping A3 is done by taking the transition function $R(\cdot, a)$ to be a relation, rather than a map, so $R(s, a) \subseteq S$. Dropping A2 is traditionally made in terms of observation functions. State-transition systems that do not assume A2-A3 have been defined with the help of the following concepts [23].

Definition B.2.3 (Non-deterministic state transition system)**.** A *non-deterministic state transition system* is a tuple $\Sigma = \langle S, A, R \rangle$, where S is a finite set of states, A is a finite set of actions, and $R : S \times A \to \mathcal{P}(S)$ is the transition function.

Dropping the A2 assumption is usually done by adding some observation function.

Definition B.2.4 (Observation function)**.** [23] Let $\Sigma = \langle S, A, R \rangle$ be a state transition system. Let $\mathcal{O}$ be a finite set of observations. An observation function over S and $\mathcal{O}$ is a function $X : S \to 2^{\mathcal{O}} \setminus \varnothing$, which associates to each state s the nonempty set of possible observations $X(s) \subseteq \mathcal{O}$.

Definition B.2.5 (Non-det. partially-observable Planning domain)**.** A *planning domain* $\mathbb{M}$ is a tuple $\langle \Sigma, \mathcal{O}, \mathcal{X} \rangle$, where: $\Sigma = \langle S, A, R \rangle$ is a non-deterministic planning domain, $\mathcal{O}$ is a finite set of observations, and $\mathcal{X}$ is an observation function over S and $\mathcal{O}$.

See [23] for a study of strong planning under partial observability, based on conditional plans.

Definition B.2.6 (Conditional Plan)**.** The set of conditional plans Plans for a domain $\langle \Sigma, \mathcal{O}, \mathcal{X} \rangle$ is the minimal set such that:

- $\varnothing \in \mathsf{Plans}$,
- if $\mathsf{e} \in \mathbf{A}$ and $\pi \in \mathsf{Plans}$, then $\pi^{\frown}(\mathsf{e}) \in \mathsf{Plans}$, and
- if $o \in \mathcal{O}$ and $\pi_1, \pi_2 \in \mathsf{Plans}$, then

$$\texttt{if}\, o\ \texttt{then}\, \pi_1, \texttt{else}\, \pi_2 \qquad \text{is in } \mathsf{Plans}$$

Comparison with LCC planning systems.

We conclude this Chapter B by relating the LCC planning system from Chapter 5 to the deterministic planning systems from the previous Section B.1, and how

does B.2 relate to the planning system from Chapter 7. We denote classical planning systems using the notation $\Sigma = (\mathbf{S}, \mathbf{A}, \mathbf{R})$ to avoid confusion.

First, note that a deterministic state transition system $\Sigma = (\mathbf{S}, \mathbf{A}, \mathbf{R})$ can be represented by a pair (U, M), where $\mathsf{U} = (\mathsf{E}, \mathsf{pre}, \mathsf{post}, \mathsf{R}_a)$ is an action model and $M = (W, R_a, V)$ is an epistemic model for the single-agent case $\mathsf{Ag} = \{a\}$.

Definition B.2.7 (Translation to LCC)**.** Let $\Sigma = \langle \mathbf{S}, \mathbf{A}, \mathbf{R} \rangle$ be a state transition system, with $\mathbf{S} = \{s_1, \ldots, s_n\}$. Let $\mathsf{Var} = \{p_1, \ldots, p_n\}$. We define the translation of Σ into LCC as a pair, a model $M = (W, R_a, V)$ and an action model $\mathsf{U} = (\mathsf{E}, \ldots)$ defined by

$$\begin{array}{lrcl} W = \{w_1, \ldots, w_n\} & \mathsf{E} & = & \{\mathsf{e}'\}_{\mathsf{e} \in \mathbf{A}} \\ R_a = Id_W & \mathsf{pre}(\mathsf{e}) & = & \bigvee \{p_i \mid \exists j (w_i \mathbf{R} w_j)\} \\ V(p_i) = \{w_i\} & \mathsf{post}(\mathsf{e})(p_j) & = & \bigvee \{p_i \mid w_i \mathbf{R} w_j)\} \\ & \mathsf{R}(\mathsf{e}, \mathsf{f}) & \text{iff} & \mathsf{f} = \mathsf{e} \end{array}$$

Proposition B.2.8. *Let $\mathbb{M} = (\Sigma, s_0, S_g)$ be a classical planning domain, with $\{s_0\} \cup S_g \subseteq S$ and let $(\mathsf{e}_1, \ldots, \mathsf{e}_n)$ be plan for $\mathbb{M}$. Then, if we let $\mathbb{M}'$ denote the* LCC *planning domain $\mathbb{M}' = (p_0, A, \bigwedge_{s_i \in S_g} p_i)$, we have*

$$(\mathsf{e}_1, \ldots, \mathsf{e}_n) \textit{ is a solution for } \mathbb{M} \quad \Rightarrow \quad (\mathsf{e}'_1, \ldots, \mathsf{e}'_n) \textit{ is a solution for } \mathbb{M}'$$

Proof. Redefine S as $S = \{s_0, s'_1, \ldots, s'_n\}$ and now let $s_0, s_1, s_2, \ldots, s_m$ be a tuple of S-states such that $\mathbf{R}(s_k, \mathsf{e}_{k+1}) = s_{k+1}$ for each $0 \leq k < m$. The proof is by induction on k for the claim

$$\mathsf{e}_{k+1} \text{ is executable in } s_k \quad \Leftrightarrow \quad \mathsf{e}'_k \text{ is executable in } w_k.$$

(Base Case) Clearly, e'_1 is executable in w_0, since $\mathsf{pre}(\mathsf{e}'_1) = p_0 \vee \ldots$ and $M, w_0 \models p_0$ so $M, w \models \mathsf{pre}(\mathsf{e}'_1)$. Thus $M \circ \mathsf{U}, (w, \mathsf{e}'_1)$ exists and moreover satisfies $M \circ \mathsf{U}, (w, \mathsf{e}'_1) \models p_1$. Since $\mathsf{post}(\mathsf{e}'_1)(p_1) = p_0 \vee \ldots$, we have that $M \circ \mathsf{U}, (w, \mathsf{e}'_1) \models \mathsf{pre}(\mathsf{e}'_2)$.
(Ind. Case) The proof is similar, using the (Ind. Hyp.) that

$$((M \circ \mathsf{U}) \cdots) \circ \mathsf{U}, ((w, \mathsf{e}'_1), \ldots, \mathsf{e}'_k) \models p_k$$

and hence this is a model for $\mathsf{pre}(\mathsf{e}'_{k+1})$ as well.

The proof concludes with the observation that $\mathsf{post}(\mathsf{e}_m) = p_m$, where $s_m \in S_g$ is a goal state. □

See [28] for a similar translation of state transition systems in the set-theoretic representation of classical planning, i.e. with a set of states initially defined as $S = \mathcal{P}(\mathsf{Var})$. The translation in [28] is made in the terms of their epistemic planning domains, where the planning actions are semantically modeled as in the action model.

Finally, the non-deterministic partially-observable planning domains $\Sigma = (\mathbf{S}, \mathbf{A}, \mathbf{R})$ of Definition B.2.5 can be translated as well into planning domains of $\mathrm{LCC}_{\cup \otimes n}$. Given a propositional representation of states $\mathbf{S} = \mathcal{P}(\mathsf{Var})$ one can

naturally identify the possible observations with states: $\mathcal{O} = \mathbf{S}$, so *an observation that p in state s* can be modeled as $\mathcal{X}(s) = [\![p]\!]^M$, for a single-agent epistemic model $M = (W, R_a, V)$. With more detail, this model M can be defined as follows: W and V are as above, and $R_a(w_i, w_j) \Leftrightarrow s_j \in \mathcal{X}(s_i)$. Non-deterministic actions $\mathsf{e} \in A$ (i.e. with $|\mathbf{R}(s, \mathsf{e})| > 1$) can again be encoded as deterministic actions in some action model U. For the case $|\mathbf{R}(s, \mathsf{e})| = 2$, for a fixed e and arbitrary states s, this can be done with two deterministic actions $\mathsf{e}'_0, \mathsf{e}'_1 \in \mathsf{E}$.

The notion of conditional plan in Definition B.2.6 is more expressive than the the $\mathbb{M}$-sequences considered in Chapter 7. See also [9] for a (forward) conditional planning system, again based on action models U, represented in a way closer to the non-deterministic partially-observable planning domains from [23].

Appendix C

Argumentation Systems

The present chapter contains an introduction to the recent area of formal and computational models of argument. It describes the relevant areas established since the publication of Dung [52]. This work proposed an abstract notion of justification or acceptability of arguments, simply represented as a set of points (or unstructured elements), possibly related to each other by a binary relation of attack.

Abstract argumentation has a natural interpretation in logic, among other areas. The concept of arguments can in particular be naturally related to that of proof, i.e. a proof in a logic, from a given set of premises or knowledge base. This logic-based interpretation provides the arguments from abstract argumentation with an internal structure, consisting of premises and inference steps (e.g. modus ponens). Following this logic-based interpretation, an attack is naturally read as involving a logical conflict between the two proofs or internal structures, e.g. some logical contradiction existing between them.

Indeed, the natural interpretation of this work [52] in terms of logic started a series of contributions in the so-called area of logic-based argumentation. See [126] for a handbook presentation of the different topics on abstract and logic-based argumentation. These logic-based approaches to argumentation, in addition, permit to focus on the acceptability of propositions, i.e. as the conclusions of acceptable arguments.

Structure of the Chapter

Section C.1, reviews the Dung acceptability semantics that define the field of abstract argumentation. In Section C.2, we briefly review the literature on logic-based argumentation, and focus on defeasible argumentation due to its relation with t-DeLP from Chapter 1. We present with some detail some of its topics, e.g. the rationality postulates, using the particular ASPIC framework.

C.1 Abstract Argumentation Frameworks

In [52], Dung introduced some formal notions of (collective) acceptability of arguments. The proposed approach is abstract in the sense that arguments are simply points a, b in a set, possibly related to each other by a binary attack relation, denoted $a \to b$, read as a is an argument against b. An argumentation system, defined by such a pair of a set of arguments and the attack relation. This abstract relation of attack can represent any form in which an argument can counter-argue another argument.

Definition C.1.1 (Argumentation system)**.** An *abstract argumentation framework* (AF) is a pair $\langle \mathbb{A}, \to \rangle$. $\mathbb{A}$ is a set arguments and $\to \subseteq \mathbb{A} \times \mathbb{A}$ is a binary relation of attack. We say that an argument a *attacks* an argument b iff $(a, b) \in \to$.

Dung semantics decide, on the sole basis of this relation, whether a particular argument is acceptable, given the attacks from other arguments, attacks on the these arguments, and so on. The different Dung-style semantics in the literature are defined in terms of properties of sets of arguments, which make them (collectively) acceptable, in the sense of: not attacking each other, and defending from external attacks –among other properties. Those sets of arguments satisfying the conditions of a given semantics are called the extensions of the semantics. Since different extensions might exist, one might consider their intersection, called the skeptical output in this semantics. These semantics are presented below. Different methods for computing these extensions have also been proposed in the literature [126].

Definition C.1.2 (Conflict-free, Defense)**.** Let $\mathbb{B} \subseteq \mathbb{A}$ be a set of arguments.

- the set $\mathbb{B}$ is *conflict-free* iff there are no a, a' in $\mathbb{B}$ such that a attacks a'.
- the set $\mathbb{B}$ defends an argument a_0 iff for each $a_1 \in \mathbb{A}$, if a_1 attacks a_0, then there is some $a_2 \in \mathbb{B}$ such that a_2 attacks a_1.

A semantics for an argumentation system is a formal method for the evaluation of arguments in an AF. Different semantics can be defined for abstract argumentation systems, in terms of the basic concept of extension [19, 126].

Definition C.1.3 (Extensions.Acceptability Semantics.)**.** For a given AF $(\mathbb{A}, \to)$, let $\mathbb{B} \subseteq \mathbb{A}$ be a conflict-free set and let $\mathcal{F} : \mathcal{P}(\mathbb{A}) \to \mathcal{P}(\mathbb{A})$ be the function $\mathcal{F}(\mathbb{B}) = \{a \in \mathbb{A} \mid \mathbb{B} \text{ defends} a\}$. We say

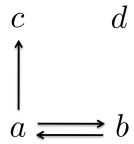
- $\mathbb{B}$ is *admissible* iff $\mathbb{B} \subseteq \mathcal{F}(\mathbb{B})$
- $\mathbb{B}$ is a *complete extension* iff $\mathbb{B} = \mathcal{F}(\mathbb{B})$
- $\mathbb{B}$ is a *grounded extension* iff it is the $\subseteq$-smallest complete extension
- $\mathbb{B}$ is a *preferred extension* iff it is a $\subseteq$-maximal complete extension
- $\mathbb{B}$ is a *stable extension* iff it is a preferred extension that attacks all arguments in $\mathbb{A} \setminus \mathbb{B}$

The set of extensions in some AF $(\mathbb{A}, \rightarrow)$ under any of these semantics $\mathcal{S} \in$ { complete, grounded, ...,} is denoted by $\mathbf{E}_{\mathcal{S}}(\mathbb{A}, \rightarrow)$.

See Figure C.1 for an illustration of these semantics. They are also called admissibility semantics, since they are all based on the property of admissible (extensions), which essentially demands internal consistency and the ability of defending from, or counter-attacking, external attacks. An extension in any of these semantics $\mathcal{S}$ can be seen as a set of collectively acceptable arguments according to $\mathcal{S}$. The different semantics describe different notions of *collective* acceptability. Some basic results about these semantics are the following:

- each arg. system has a grounded extension,
- each arg. system has at least one preferred extension, and one complete extension
- some arg. systems exist without stable extensions

In [52] the grounded semantics has been shown to correspond to the well-founded semantics of logic programs [148].



Conflict-free:	$\{a\}, \{b\}, \{c\}, \{d\}$, $\{a, d\}, \{b, d\}, \{c, d\}$, $\{b, c\}, \{b, c, d\}$
Defense:	$\{b\}$ defends b and c, etc.
Extensions	
Admissible:	$\{a\}, \{b\}, \{d\}$, $\{a, d\}, \{b, c\}\{b, d\}$, $\{b, c, d\}$
Complete:	all the admissible except for $\{b\}, \{b, d\}$
Grounded:	$\varnothing$
Preferred:	$\{a, d\}, \{b, c, d\}$
Stable:	$\{a, d\}, \{b, c, d\}$

Figure C.1: (Left) An example of an argumentation framework, with arguments $\{a, \ldots, d\}$ and the attack relation represented by arrows. (Right) Lists of extensions under the different semantics.

Different extensions of abstract argumentation systems have been considered with further notions, also at an abstract level, see [126]. Among these, one can include: preferences, values, probabilities, supports between arguments, as well as sets of attacks [101].

C.2 Logic-based argumentation

As explained above, the abstract argumentation systems neither do account for the nature or origin of arguments nor for the attack relation assumed to hold between them. From the point of view of a given logic (with negation), in contrast, these notions have (at least) a natural interpretation:

$\mathcal{A}$ is an argument	iff	$\mathcal{A}$ is a consistent logical derivation (from a set of premises or non-logical axioms)
$\mathcal{A}$ attacks $\mathcal{B}$	iff	the conclusion (or any sub-conclusion) of $\mathcal{A}$ contradicts some step used by $\mathcal{B}$

Different logics have been studied as providing a logical foundation for argumentation systems. For instance, arguments expressible in classical logic [24], general logical languages [121] (including [149, 118, 119]), rule-based systems [35, 122, 36] and logic programming [60, 61].

The above natural reading of attack in a given logic (only) captures the existence of a symmetric conflict. This has led to more refined notions, called *defeat*, in order to account for some asymmetries between two conflicting arguments. Using the notation from Chapter 1, from here on we write instead an argument system as a pair $(\mathbb{A}, \succ)$ rather than $(\mathbb{A}, \rightarrow)$.

A first decision in the design of a framework for logic-based argumentation in a selected logic is to define which are the specific targets of attacks within attacked or defeated arguments. This is partly determined by the base logic, though. For example, in argumentation based on classical logic, the premises are natural objects of attack by arguments concluding facts contradicting these premises. In rule-based systems, the defeasible rules can be considered as the objects of the attacks in the corresponding argumentation systems.

In the following, we illustrate some basic concepts and topics in logic-based argumentation using a particular rule-based system ASPIC.[1] Some technical details on the construction of arguments, though, are not provided, for the sake of simplicity; see [6] and [35] for details. These arguments are defined with the help of the strict and defeasible rules of a defeasible theory.

Definition C.2.1 (Defeasible theory)**.** For a given language of literals, a *defeasible theory* is a pair $\langle \mathbf{S}, \mathbf{D} \rangle$ where $\mathbf{S}$ is a set of strict rules (denoted with $\rightarrow$), and $\mathbf{D}$ is a set of defeasible rules (denoted with $\Rightarrow$).

Definition C.2.2 (Closure; Consistency)**.** Let $\langle \mathbf{S}, \mathbf{D} \rangle$ be a defeasible theory. The *closure under* $\mathbf{S}$ of some set of literals $\mathcal{P}$ is defined as usual as the closure under modus ponens with rules from $\mathbf{S}$. The set resulting from this closure is denoted $\mathrm{Cn}_{\mathbf{S}}(\mathcal{P})$, or simply $\mathrm{Cn}(\mathcal{P})$. If we denote strong negation as $\sim$, a set $\mathcal{P}$ is consistent iff no pair of the form $\ell, \sim\ell$ exists in $\mathcal{P}$.

As usual, it its assumed that the set $\mathbf{S}$ is such that $\mathrm{Cn}_{\mathbf{S}}(\varnothing)$ consistent. For defeasible theories, the constructible arguments are minimal consistent proofs

[1] We present the version studied in [35] for illustrative purposes; newer versions of this system have been later adopted.

built from $\mathbf{S} \cup \mathbf{D}$ by successive applications of modus ponens. The set of arguments $\mathcal{A}$ constructible from a defeasible theory is again denoted $\mathbb{A}$. The set of sub-arguments of an argument $\mathcal{A}$ is denoted $\mathsf{Sub}(\mathcal{A})$. We use of the notation $\mathsf{concl}(\cdot)$ for the conclusion of an argument.

Besides the above constraints imposed by a logic upon the objects of attack in the corresponding logic-based argumentation system, an attack (later called defeat) can consist in either of the two types

- a logical conflict or inconsistency, called *rebuttal*; or
- an argument concluding the inapplicability of one of the rules used by the attacked argument; this is called *undercut*

Example C.2.3. [35] Consider an argument $\mathcal{A}$ = "The object is red because John says it looks red". A rebutter of $\mathcal{A}$ can be $\mathcal{B}_1$ = "The object is not red because Suzy says it looks blue". An undercutter of $\mathcal{A}$ can be $\mathcal{B}_2$ = "The object is merely illuminated by a red light." The latter argument $\mathcal{B}_2$ is not a reason against the claim that the object is red, only against the fact that $\mathcal{A}$ is a good argument for this claim: that it looks red is no longer a reason for its being actually red.

Undercuts are defined by extending the language with new literals. Thus, if δ is a defeasible rule in $\mathbf{D}$, the language is assumed to contain a new literal $\ulcorner\delta\urcorner$. This is the possible target of an undercutting argument, whose conclusion is $\sim\ulcorner\delta\urcorner$.

Definition C.2.4 (Rebut; Undercut; Defeat)**.** Given two arguments constructible from a defeasible theory $\langle \mathbf{S}, \mathbf{D}\rangle$, we say that

- $\mathcal{A}$ rebuts $\mathcal{B}$, if there exist $\mathcal{A}' \in \mathsf{Sub}(\mathcal{A})$ and $\mathcal{B}' \in \mathsf{Sub}(\mathcal{B})$ such that $\mathsf{concl}(\mathcal{A}) = \sim\mathsf{concl}(\mathcal{B})$ and the rule for $\mathsf{concl}(\mathcal{B}')$ is in $\mathbf{D}$;
- $\mathcal{A}$ undercuts $\mathcal{B}$, if $\mathsf{concl}(\mathcal{A}) = \sim\ulcorner\delta\urcorner$ for some $\delta \in \mathcal{B} \cap \mathbf{D}$.

We say $\mathcal{A}$ defeats $\mathcal{B}$, denoted $\mathcal{A} \succ \mathcal{B}$ iff $\mathcal{A}$ rebuts or undercuts $\mathcal{B}$.

The above notion of rebut is called restricted rebut in [35]. The above Dung semantics from SectionC.1 can be applied to argumentation systems $(\mathbb{A}, \succ)$ induced by a given defeasible theory $\langle \mathcal{S}, \mathcal{D}\rangle$. Since these semantics are only defined for sets of arguments (extensions), the set of collectively acceptable conclusions from a defeasible theory or argumentation system must also be defined.

Definition C.2.5 (Conclusions; Output)**.** Let $\langle \mathbb{A}, \succ\rangle$ be an argumentation system, and let $\{E_1, \ldots, E_n\}$ be its set of extensions under one of the semantics $\mathcal{S}$ from Def. C.1.3. We define:

- $\mathsf{Concs}(E_i) = \{\mathsf{concl}(\mathcal{A}) \mid \mathcal{A} \in E_i\}$, for each $1 \leq i \leq n$.
- $\mathsf{Output} = \bigcap_{1\leq i\leq n} \mathsf{Concs}(E_i)$

This kind of outputs are called skeptical. Credible outputs simply consist in picking the conclusions of some particular extension(s).

In [35], Caminada and Amgoud proposed a list of desirable properties that argumentation systems should satisfy. See also [121]. These properties, called Rationality Postulates, are defined as follows.

Definition C.2.6 (Rationality Postulates)**.** Let $\langle \mathbb{A}, \succ \rangle$ be the argumentation system induced by some defeasible theory. And let $\{E_1, \ldots, E_n\}$ be the set of its extensions under $\mathcal{S}$. We define the *rationality postulates* as the following conditions:

(Sub-arguments) We say that $(\mathbb{A}, \succ)$ satisfies *closure under sub-arguments* iff

- $\mathsf{Sub}[E_i] \subseteq E_i$

(Closure) We say that $(\mathbb{A}, \succ)$ satisfies *closure* iff

- $\mathsf{Concs}(E_i) = \mathrm{Cn}_{\mathcal{S}}(\mathsf{Concs}(E_i))$, for each $1 \leq i \leq n$
- $\mathsf{Output} = \mathrm{Cn}_{\mathcal{S}}(\mathsf{Output})$

(Direct Consistency) We say that $(\mathbb{A}, \succ)$ satisfies *direct consistency* iff

- $\mathsf{Concs}(E_i)$ is consistent, for each $1 \leq i \leq n$
- Output is consistent

(Indirect Consistency) We say that $(\mathbb{A}, \succ)$ satisfies *indirect consistency* iff

- $\mathrm{Cn}_{\mathcal{S}}(\mathsf{Concs}(E_i))$ is consistent, for each $1 \leq i \leq n$
- $\mathrm{Cn}_{\mathcal{S}}(\mathsf{Output})$ is consistent

Basic relations among these postulates are the following: indirect consistency implies direct consistency; and direct consistency and closure imply indirect consistency.

Counter-examples to the postulates of closure and indirect consistency have been found in argumentation systems from the literature on logic-based argumentation. These systems include the ASPIC and DeLP systems, as well as others [6, 61, 10, 122]. The authors of [35] propose a condition that suffices to prevent these kinds of counter-examples in ASPIC, and suggest this condition should suffice as well for the other systems.

Definition C.2.7 (Transposition)**.** A *transposition* of a strict rule δ is defined as any replacement of the form

$$\begin{array}{c} \ell_1, \ldots, \ell_i, \ldots \ell_n \ \rightarrow \ \ell \\ \Downarrow \\ \ell_1, \ldots, \ell, \ldots \ell_n \ \rightarrow \ \ell_i \end{array}$$

for an arbitrary $1 \leq i \leq n$. The set of transpositions of δ is denoted $\mathsf{tp}(\delta)$. We say $\mathbf{S}$ is closed under transpositions if $\mathsf{tp}[\mathbf{S}] \subseteq \mathbf{S}$.

The condition that $\mathbf{S}$ is closed under transpositions suffices for the satisfaction of the rationality postulates in the argumentation system defined in this section.

Theorem C.2.8. *[35] For any $\langle \mathbf{S}, \mathbf{D} \rangle$ is a defeasible theory with a consistent set of strict rules $\mathrm{Cn}_{\mathbf{S}}(\varnothing)$, and any of the semantics $\mathcal{S}$ from Def. C.1.3, it holds that*

$$\textit{if } \mathbf{S} \subseteq \mathsf{tp}(\mathbf{S}), \quad \textit{then } \langle \mathbb{A}, \succ \rangle \textit{ satisfies the rationality postulates}$$

Bibliography

[1] M. Abadi and Z. Manna. Temporal logic programming. In *Proc. of International Symposium on Logic Programming*, pages 4–16, 1987. 6, 7, 175

[2] T. Ågotnes and H. van Ditmarsch. What will they say?public announcement games. *Synthese*, 179:57–85, 2011. 121

[3] T. Alsinet, R. Béjar, and L. Godo. A characterization of collective conflict for defeasible argumentation. In *Proceedings of 3rd Computational Models of Argument COMMA 2010*, pages 27–38, 2010. 40

[4] T. Alsinet, R. Béjar, L. Godo, and F. Guitart. Maximal ideal recursive semantics for defeasible argumentation. In *Proc. of 5th Scalable Uncertainty Management SUM 2011*, pages 96–109. LNAI 6929 Springer, 2011. 40

[5] L. Amgoud. A formal framework for handling conflicting desires. In *Proceedings of Symbolic and Quantitative Approaches to Reasoning and Uncertainty, 7th European Conference, ECSQARU 2003*, pages 552–563. LNAI 2711 Springer, 2003. 99

[6] L. Amgoud, M. Caminada, C. Cayrol, M. Lagasquie, and H. Prakken. Towards a consensual formal model: inference part (technical report) deliverable d2.2. 2004. 206, 208

[7] L. Amgoud and C. Cayrol. Inferring from inconsistency in preference-based argumentation frameworks. *International Journal of Automated Reasoning*, 29(2):125–169, 2002. 40

[8] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34:197–215, 2002. 39

[9] M. Andersen, T. Bolander, and M. Jensen. Conditional epistemic planning. In *Proceedings of 13th European Conference on Logics in Artificial Intelligence JELIA 2012*, pages 94–106. LNAI 7519 Springer, 2012. 150, 169, 177, 201

[10] G. Antoniou, D. Billington, G. Governatori, and M. Maher. A flexible framework for defeasible logics. In *Proceedings of the 17th National Conference on Artificial Intelligence AAAI 2000*, pages 401–405. AAAI/MIT Press, 2000. xvi, 208

[11] G. Aucher. An internal version of epistemic logic. *Studia Logica*, 94:1–22, 2010. 124

[12] G. Aucher. Del-sequents for progression. *Journal of Applied Non-Classical Logics*, 21(3-4):289–321, 2011. 131, 132, 133

[13] G. Aucher and T. Bolander. Undecidability in epistemic planning. In *Proceedings of 23rd Int. Joint Conference on Artificial Intelligence IJCAI 2013*. IJCAI/AAAI, 2013. 132

[14] J. Augusto and G. Simari. Temporal defeasible reasoning. *Knowledge and Information Systems*, 3:287–318, 2001. 8, 35, 37, 38, 40, 49

[15] J. Austin. *How To Do Things with Words*. Oxford University Press, 1962. 130

[16] A. Baltag, B. Coecke, and M. Sadrzadeh. Epistemic actions as resources. *Journal of Logic and Computation*, 17:555–585, 2007. 121, 133

[17] A. Baltag and L. Moss. Logic for epistemic programs. *Synthese*, 139:165–224, 2004. 112, 121, 133, 147

[18] A. Baltag, L. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicions. In *Proceedings of 7th Theoretical Aspects of Rationality and Knowledge TARK 98*, pages 43–56, 1998. 112, 120, 121, 147

[19] P. Baroni and M. Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, 171(10,15):675–700, 2007. 204

[20] M. Baudinet. On the expressiveness of temporal logic programming. *Information and Computation*, 117(2):157–180, 1995. 6, 7, 39, 175

[21] A. Belesiotis, M. Rovatsos, and I. Rahwan. Agreeing on plans through iterated disputes. In *Proceedings of 9th Conference on Autonomous Agents and MultiAgent Systems AAMAS 2010*, pages 765–772, 2010. 99

[22] N. Belnap, M. Perloff, and M. Xu. *Facing the future. Agents and Choices in Our Indeterminst World*. Oxford University Press, 2001. 121, 133

[23] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Strong planning under partial observability. *Artificial Intelligence*, 170:337–384, 2006. 168, 199, 201

[24] P. Besnard and A. Hunter. Argumentation based on classical logic. In Rahwan and Simari, editors, *Argumentation in Artificial Intelligence*, chapter 7. Springer, 2009. 206

[25] D. Billington. Defeasible logic is stable. *Journal of Logic and Computation*, 3:379–400, 2993. 40

[26] P. Blackburn, J. van Benthem, and F. Wolter (eds.). *Handbook of Modal Logic*. Elsevier, 2006. 38

[27] A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997. 197

[28] T. Bolander and M. Andersen. Epistemic planning for single- and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011. xvi, 132, 152, 153, 169, 177, 200

[29] R. Brafman, C. Domshlak, Y. Engel, and M. Tennenholtz. Planning games. In *Proceedings of International Joint Conference on Artificial Intelligence IJCAI 2009*, pages 73–78, 2009. 100

[30] M. Bratman. *Intentions, Plans and Practical Reason*. Harvard University Press, 1987. 133

[31] P. Bretier and D. Sadek. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. In *Intelligent Agents III*, pages 189–204. Springer, 1997. 130

[32] G. Brewka, I. Niemelä, and M. Truszczyński. Nonmonotonic reasoning. In van Harmelen, Lifschitz, and Porter, editors, *Handbook of Knowledge Representation*, chapter 6. Elsevier, 2007. 39

[33] J. Broersen, R. Wieringa, and J.-J. Meyer. A semantics for persistency in propositional dynamic logic. In *Proceedings of 1st Conference on Computation Logic CL 2000*, pages 912–925. Springer, 2000. 40

[34] P. Buzing, A. ter Mors, J. Valk, and C. Witteveen. Coordinating self-interested planning agents. *Autonomous Agent and Multi-Agent Systems*, 12:199–218, 2006. 99

[35] M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171:286–310, 2007. 9, 23, 26, 39, 40, 206, 207, 208, 209

[36] M. Capobianco, C. Ches nevar, and G. Simari. Argumentation and the dynamics of warranted beliefs in changing environments. *Autonomous Agents and Multi-Agent Systems*, 11(2):127–151, 2005. 11, 40, 206

[37] A. Casali, L. Godo, and C. Sierra. A graded bdi agent model to represent and reason about preferences. *Artificial Intelligence*, 175(7-8):1468–1478, 2011. xvi

[38] M. Castilho, O. Gasquet, and A. Herzig. Formalizing action and change in modal logic i: the frame problem. *Journal of Logic and Computation*, 9(5):701–735, 1999. 39

[39] L. Cecchi, P. Fillottrani, and G. Simari. On the complexity of delp through game semantics. In *Proceedings of Non-Monotonic Reasoning 2006*, pages 386–394, 2006. 175

[40] B. Chellas. *Modal logic, an introduction.* Cambridge University Press, 1980. 38, 108

[41] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147:35–84, 2003. 168

[42] L. Cobo, D. Martínez, and G. Simari. Acceptability in timed frameworks with intermittent arguments. In *Proceedings of Artificial Intelligence Applications and Innovations AIAI 2011, Part II*, pages 202–211. Springer, 2011. 40

[43] L. Cobo, D. Martínez, and G. Simari. Stable extensions in timed argumentation frameworks. In *Proceedings of Theories and Applications of Formal Argumentation TAFA 2011*, pages 181–196. Springer, 2011. 37, 40

[44] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3):213–261, 1990. 133

[45] P. Cohen and H. Levesque. Rational interaction as the basis for communication. In *Intentions in Communication*, pages 221–255. MIT Press, 1990. 130

[46] P. Cohen and C. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177–212, 1979. 130

[47] R. Craven and M. Sergot. Distant causation in $\mathcal{C}+$. *Studia Logica*, 79:73–96, 2005. 40

[48] E. Davis and L. Morgenstern. A first-order theory of communication and multi-agent plans. *Journal of Logic and Computation*, 15(5):701–749, 2005. 100

[49] T. de Lima. *Optimal Methods for Reasoning about Actions and Plans in Multi-Agents Systems. Ph.D. thesis.* IRIT, University of Toulouse 3, 2007. 169

[50] T. Delladio and G. Simari. Relating delp and default logic. *Inteligencia Artificial*, 35:101–109, 2007. 40, 49

[51] S. Demri and E. Orłowska. Logical analysis of demonic nondeterministic programs. *Theoretical Computer Science*, 166:173–202, 1996. 150

[52] P. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games 1. *Artificial Intelligence*, 77(2):321–357, 1995. 5, 8, 23, 35, 39, 40, 99, 203, 204, 205

[53] P. Economou. Sharing beliefs about actions: A parallel composition operator for epistemic programs. In *Proceedings of the ESSLLI 2005 Belief Revision and Dynamic Logic workshop*, 2005. 150

[54] G. Weiss (ed.). *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999. xiv

[55] E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 996–1072. Elsevier, 1990. 38

[56] E. Emerson and J. Srinivasan. *Branching time temporal Logic*. Springer, 1989. 38, 82, 133

[57] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995. 120, 124

[58] T. French and H. van Ditmarsch. Undecidability for arbitrary public announcement logic. In *Advances in Modal Logic AiML 2008*, pages 3–42, 2008. 121

[59] M. Maher G. Antoniou and D. Billington. Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming*, 42(1):47–57, 2000. 40

[60] A. García, J. Dix, and G. Simari. Argument-based logic programming. In Rahwan and Simari, editors, *Argumentation in Artificial Intelligence*, chapter 8. Springer, 2011. 206

[61] A. García and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1+2):95–138, 2004. xvi, xxi, 5, 7, 8, 15, 19, 20, 35, 36, 39, 40, 206, 208

[62] D. García, A. García, and G. Simari. Defeasible reasoning and partial order planning. In *Proceedings of the 5th Foundations of Information and Knowledge Systems FoIKS 2008*, pages 311–328. LNCS 4932 Springer, 2008. xxi, 81, 176

[63] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17(2,3&4):301–321, 1993. 40

[64] J. Gerbrandy. Logics of propositional control. In *Proceedings of Autonomous Agents and Multiagent Systems AAMAS 2006*, pages 193–200. IFAAMAS, 2006. 121

[65] J. Gerbrandy and W. Groenevelt. Reasoning about information change. *Journal of Logic, Language and Information*, 6(2):147–169, 1997. 111, 120, 121

[66] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004. 48, 49, 50, 82, 194, 196

[67] G. De Giacomo and M. Lenzerini. Pdl-based framework for reasoning about actions. In *Proceedings of the 4th Congress of the Italian Association for Artificial Intelligence IA*AI95*, pages 103–114. LNAI 992 Springer, 1995. 39

[68] L. Giordano, A. Martelli, and C. Schwind. Ramification and causality in a modal action logic. *Journal of Logic and Computation*, 10(5):625–662, 2000. 40

[69] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153:49–104, 2004. 40

[70] L. Godo, E. Marchioni, and P. Pardo. Extending a temporal defeasible argumentation framework with possibilistic weights. In *Proceedings of 13th European Conference on Logics in Artificial Intelligence JELIA 2012*, pages 242–254. LNAI 7519 Springer, 2012. xvii, xx, 40

[71] R. Goldblatt. *Logics of time and computation*. CSLI, 1992. 38

[72] G. Governatori and P. Terenziani. Temporal extensions to defeasible logic. In *Proceedings of 20th Australian Joint Conference on Artificial Intelligence AI 2007*, pages 1–10. Springer, 2007. 40, 49

[73] B. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(269–357), 1996. 100

[74] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000. 38

[75] A. Herzig, J. Lang, D. Longin, and T. Polacsek. A logic for planning under partial observability. In *Proceedings of the 17th National Conference on Artificial Intelligence AAAI 2000*, pages 768–773. AAAI/MIT Press, 2000. 133

[76] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962. 108, 120

[77] W. Holliday, T. Hoshi, and T. Icard. Schematic validity in dynamic epistemic logic: Decidability. In *Proceedings of Logics of Rational Interaction LORI 2011*, pages 87–96. LNAI 6953 Springer, 2011. 112

[78] J. Hughes, P. Kroes, and S. Zwart. A semantics for means-end relations. *Synthese*, 158:207–231, 2007. xiv

[79] J. Hulstijn and L. van der Torre. Combining goal generation and planning in an argumentation framework. In *Proceedings of Non-Monotonic Reasoning NMR 2004 Workshop on Argument, Dialogue and Decision*, pages 212–218, 2004. 99

[80] A. Hunter. Execution of defeasible temporal clauses for building preferred models. In *Proceedings of Fundamentals of Artificial Intelligence Research FAIR '91*, pages 84–98. Springer, 1991. 40

[81] A. Hunter. Merging structured text using temporal knowledge. *Data Knowledge Engineering*, 41(1):29–66, 2002. 40

[82] A. Jonsson and M. Rovatsos. Scaling up multiagent planning: A best-response approach. In *Proceedings of 21st Automated Planning and Scheduling ICAPS 2011*. AAAI 2011, 2011. 99

[83] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35:342–382, 1988. 40

[84] B. Kooi. Probabilistic dynamic epistemic logic. *Journal of Logic, Language and Information*, 12:381–408, 2003. 121

[85] B. Kooi. Expressivity and completeness for public update logics via reduction axioms. *Journal of Applied Non-Classical Logics*, 17(2):231–253, 2007. 112, 121

[86] B. Kooi and B. Renne. Arrow update logic. *Review of symbolic logic*, 4(4):536–559, 2011. 121

[87] B. Kooi and B. Renne. Generalized arrow update logic. In *Proceedings of Theoretical Aspects of Rationality and Knowledge*, pages 205–211. ACM, 2011. 121

[88] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986. 40

[89] R. Ben Larbi, S. Konieczny, and P. Marquis. Extending classical planning to the multi-agent case: A game-theoretic approach. In *Proceedings of Symbolic and Quantitative Approaches to Reasoning and Uncertainty, 9th European Conference, ECSQARU 2007*, pages 731–742. LNAI 4724 Springer, 2007. 100

[90] B. Van Linder. *Modal Logic for Rational Agents. Ph.D. thesis.* Utrecht University, 1996. xvi

[91] J. Lloyd. *Foundations of Logic Programming.* Springer, 1993. 6

[92] A. Lomuscio. *Information Sharing Among Ideal Agents. Ph.D. thesis.* University of Birmingham, 1999. xvi

[93] E. Lorini. A dynamic logic of knowledge, graded beliefs and graded goals and its application to emotion modeling. In *Proceedings of the Workshop on Logic, Rationality and Interaction LORI 2011*, pages 165–178. LNAI 6953 Springer, 2011. 121

[94] B. Löwe, E. Pacuit, and A. Witzel. Del planning and some tractable cases. In *Proceedings of Logics of Rational Interaction LORI 2011*, pages 179–192. LNAI 6953 Springer, 2010. 132

[95] N. Mann and A. Hunter. Argumentation using temporal knowledge. In *Proc. of Computer Models of Argumentation COMMA 2008*, pages 204–215. IOS Press, 2008. 37, 40, 49

[96] R. Mattmüller and J. Rintanen. Planning for temporally extended goals as propositional satisfiability. In *International Joint Conference on Artificial Intelligence IJCAI 2007*, pages 1966–1972, 2007. 82

[97] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969. 39

[98] J. Miller and L. Moss. The undecidability of iterated modal relativization. *Studia Logica*, 79:373–407, 2005. 150

[99] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173:901–934, 2009. 39

[100] C. Ches nevar, J. Dix, F. Stolzenburg, and G. Simari. Relating defeasible and normal logic programming through transformation properties. *Theoretical Computer Science*, 290:499–529, 2003. 40

[101] S.H. Nielsen and S. Parsons. A generalization of dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *Proceedings of ArgMAS 2006*, pages 54–73, 2006. 205

[102] D. Nute. Defeasible logic. In Gabbay, Hogger, and Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, chapter 3, pages 353–395. Oxford University Press, 1994. xvi, 40, 133

[103] V. Padmanabhan. *On Extending BDI Logics. Ph.D. thesis.* Griffith University, 2003. xv, xvi, 133

[104] S. Pajares-Ferrando and E. Onaindia. Defeasible argumentation for multiagent planning in ambient intelligence applications. In *Proceedings of 11th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2012*, pages 509–516, 2012. 100

[105] P. Panangaden and M. Sadrzadeh. Learning in a changing world via algebraic modal logic. In *Proceedings of Algebraic Methodology and Software Technology AMAST 2010*, pages 128–141, 2010. 121

[106] P. Pardo and L. Godo. t-delp: a temporal extension of the defeasible logic programming argumentative framework. In *Proc. of Scalable Uncertainty Management SUM 2011*, pages 489–503. LNAI 6929 Springer, 2011. xvii, xx

[107] P. Pardo and L. Godo. An argumentation-based multi-agent temporal planning system built on t-delp. In *Proceedings of the Spanish Conference on Artificial Intelligence CAEPIA 2013*, (In Press). xviii, xxi

[108] P. Pardo and L. Godo. t-delp: an argumentation-based temporal defeasible logic programming framework. *Annals of Mathematics and Artificial Intelligence*, (In press). xvii, xx

[109] P. Pardo and L. Godo. A temporal argumentation approach to cooperative planning using dialogues. In *Proceedings of the 14th Workshop on Computational Logic in Multi-Agent Systems CLIMA 2013*, (In Press). xviii, xxi

[110] P. Pardo, S. Pajares, E. Onaindia, L. Godo, and P. Dellunde. Multiagent argumentation for cooperative planning in delp-pop. In *Proc. of Autonomous Agents and Multi-Agent Systems AAMAS 2011*, pages 971–978. IFAAMAS, 2011. xviii, xxi, xxii, 100

[111] P. Pardo, S. Pajares, E. Onaindia, L. Godo, and P. Dellunde. Cooperative dialogues for defeasible argumentation-based planning. In *Proceedings of Argumentation in Multi-Agent Systems ArgMAS 2011*, pages 174–193. LNAI 7543 Springer, 2012. xviii, xxii, 82, 100, 176

[112] P. Pardo and M. Sadrzadeh. Backward planning in the logics of communication and change. In *Proceedings of 1st International Conference on Agreement Technologies AT 2012*, pages 231–245. CEUR, 2012. xix, xxii

[113] P. Pardo and M. Sadrzadeh. Planning in the logics of communication and change. In *Proceedings of Autonomous Agents and Multi-Agent Systems AAMAS 2012*, pages 1231–1232. IFAAMAS, 2012. xviii, xxii

[114] P. Pardo and M. Sadrzadeh. Strong planning in the logics of communication and change. In *Post-Proceedings of Declarative Agents, Languages and Technologies DALT 2012*, pages 37–56. Springer, 2012. xix, xxii

[115] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984. 185, 186, 192

[116] J. Penberthy and D. Weld. Ucpop: A sound, complete, partial order planner for adl. In *Proc. of 3rd Int. Conf. on Knowledge Representation and Reasoning KR'92*, pages 103–114, 1992. 197

[117] J. Plaza. Logics of public communications. In *Proceedings of 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989. 111, 112, 120, 121

[118] J. Pollock. Defeasible reasoning. *Cognitive Science*, 11:481–518, 1987. 206

[119] J. Pollock. Justification and defeat. *Artificial Intelligence*, 67:377–408, 1994. 206

[120] D. Poole. On the comparison of theories: Preferring the most specific explanation. In *Proceedings of 9th International Joint Conference in Artificial Intelligence IJCAI'85*, pages 144–147. Morgan-Kaufmann, 1985. 7, 40

[121] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010. 5, 23, 39, 40, 206, 208

[122] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7:25–75, 1997. 39, 206, 208

[123] H. Prendinger and G. Schurz. Reasoning about action and change: a dynamic logic approach. *Journal of Logic, Language, and Information*, 5:209–245, 1996. 39

[124] I. Rahwan and L. Amgoud. An argumentation-based approach for practical reasoning. In *Proceedings of 5th Conference on Autonomous Agents and Multi-Agent Systems AAMAS 2006*, pages 347–354, 2006. 99

[125] I. Rahwan and L. Amgoud. An argumentationbased approach for practical reasoning. In *Proceedings of Autonomous Agents and Multi-Agent Systems AAMAS 2006*, pages 347–354. IFAAMAS, 2006. xvi

[126] I. Rahwan and G. Simari (eds.). *Argumentation in Artificial Intelligence*. Springer, 2011. 39, 99, 203, 204, 205

[127] A. Rao and M. Georgeff. Modelling rational agents within a bdi-architecture. In *Proceedings of Principles of Knowledge Representation and Reasoning KR91*, pages 473–484. Morgan Kaufmann, 1991. 133

[128] A. Rao and M. Georgeff. Decision procedures for bdi logics. *Journal of Logic and Computation*, 8:293–342, 1998. 133

[129] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980. 40

[130] J. Rintanten. On specificity in default logic. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence IJCAI 95*, pages 1474–1479. Morgan-Kaufmann, 1995. 40

[131] T. Sadzik. *Exploring the iterated update universe (technical report)*. ILLC, University of Amsterdam, 2006. 150

[132] R. Schmidt, D. Tishkovsky, and U. Hustadt. Interactions between knowledge, action and commitment within agent dynamic logic. *Studia Logica*, 78:381–415, 2004. 150

[133] J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969. 130

[134] F. Stolzenburg, A. García, C. Ches nevar, and G. Simari. Computing generalized specificity. *Journal of Applied Non-Classical Logics*, 12(1):1–27, 2002. 7, 40

[135] M. Thimm. Realizing argumentation in multi-agent systems using defeasible logic. In *Proceedings of Argumentation in Multi-Agent Systems ArgMAS 2009*, pages 175–194. LNAI 6057 Springer, 2010. 85, 99, 100, 175

[136] M. Thimm and G. Kern-Isberner. On the relationship of defeasible argumentation and answer set programming. In *Proceedings of Computer Models of Argumentation COMMA'08*, pages 393–404. IOS Press, 2008. 26, 40

[137] J. van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155, 2007. 177

[138] J. van Benthem, J. Gerbrandy, and B. Kooi. Dynamic update with probabilities. *Studia Logica*, 93(1):67–96, 2009. 121

[139] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204:1620–1662, 2006. xvii, xviii, xix, xxii, 105, 107, 108, 111, 112, 113, 116, 117, 119, 121, 126, 139, 144, 147, 150, 176

[140] W. van der Hoek, B. van Linder, and J.-J. Meyer. On agents that have the ability to choose. *Studia Logica*, 65:79–119, 2000. 146, 150

[141] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proc. of Autonomous Agents and Multi-Agent Systems AAMAS 2002*, pages 1167–1174. IFAAMAS, 2002. xvi, 132

[142] H. van Ditmarsch and B. Kooi. Semantic results for ontic and epistemic change. In *Proceedings of Logic and the Foundations of Game and Decision Theory LOFT*, pages 87–117. Amsterdam University Press, 2008. 112, 116, 133, 150, 169

[143] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Concurrent dynamic epistemic logic for mas. In *Proceedings of Autonomous Agents and Multi-Agent Systems AAMAS 2003*, pages 201–208. ACM Press, 2003. 150, 177

[144] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. In *Proc. of Autonomous Agents and Multiagent Systems AAMAS 2005*, pages 955–960. ACM, 2005. 121

[145] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Springer, 2008. 112, 120, 132

[146] J. van Eijck and F. Sietsma. Multi-agent belief revision with linked preferences. In *Proceedings of Logic and the Foundations of Game and Decision Theory LOFT*, pages 174–189. Springer, 2008. 177

[147] J. van Eijck and Y. Wang. Propositional dynamic logic as a logic of belief revision. In *Proceedings of Workshop on Logic Language Information and Computation WoLLIC 2008*, pages 136–148. LNAI 5110 Springer, 2008. 177

[148] A. van Gelder, K. Ross, and J. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM Association for Computing Machinery*, 38(3):620–650, 1991. 205

[149] G. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90:225–279, 1997. 206

[150] H. Wansing. Tableaux for multi-agent deliberative-stit logic. In *Advances in Modal Logic AiML 2006*, pages 503–520, 2006. 133

[151] M. Wooldridge and N. Jennings (eds.). *Intelligent Agents*. Springer, 1995. xiv, xvi

[152] D. Zhang and N. Foo. Frame problem in dynamic logic. *Journal of Applied Non-Classical Logics*, 15(2):215–239, 2005. 39, 40, 82