PhD-FSTM-2024-057
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 04/07/2024 in Bologna

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

AND

DOTTORE DI RICERCA

IN LAW, SCIENCE AND TECHNOLOGY

by

Giuseppe PISANO
Born on 3 September 1995 in Italy

ARGUMENTATION FOR LEGAL REASONING: META-MODELS, TECHNOLOGY AND BEYOND

Dissertation defence committee

Dr Leon van der Torre, dissertation supervisor
Professor, Université du Luxembourg

Dr Andrea Omicini, dissertation supervisor
Professor, Università di Bologna

Dr Emiliano Lorini, Chairman
Professor, Université de Toulouse

Dr Dino Pedreschi
Professor, Università di Pisa

Dr Giovanni Sileno
Professor, University of Amsterdam

Abstract

This thesis presents a comprehensive exploration of *argumentation* in the context of legal reasoning, bridging the gap between formal argumentation theory and its technological applications. Central to this work is the enhancement of the ASPIC$^+$ framework, integrating structured meta-argumentation to address limitations in reasoning about rules, conflicts, and preferences, including the concept of the burden of persuasion. This advancement expands the framework's applicability in *legal reasoning* and beyond.

A pivotal aspect of this research is the development of the Arg2P framework, a robust and versatile environment integrating theoretical advancements in argumentation. The framework marks a significant stride in realising practical, logic-based environments for argumentation in intelligent systems, demonstrating a marked focus on user-friendliness and technical maturity, crucial for bridging theoretical innovation with functional application.

The thesis also delves into the realm of *machine learning* (ML), illustrating the integration of structured argumentation with *automated machine learning* (AutoML). This integration is aimed at enhancing the transparency and control in the development of ML systems by offering a symbolic interface for incorporating expertise in ML, exemplifying the convergence of traditional *symbolic AI* methods with data-driven ML approaches.

This work significantly contributes to argumentation theory and legal AI, providing a nuanced understanding of *meta-argumentation* and its practical applications. The enhancements to ASPIC$^+$, coupled with the Arg2P framework, present new avenues for legal analysis and decision-making. The integration with ML further highlights the potential of structured argumentation in contemporary AI, paving the way for more robust and ethically sound AI systems across various domains.

Acknowledgements

The work has been supported by the "CompuLaw" project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 833647).

Contents

List of Figures

Listings

Chapter 1

Introduction

In today's technological panorama, characterised by rapid evolution and interdisciplinary integration, this thesis presents a work at the intersection of formal argumentation, machine learning (ML) and legal reasoning. These fields come together to address some of the most pressing challenges and opportunities in the current technological panorama. Formal argumentation provides a structured approach to dissecting and constructing logical arguments, a cornerstone in AI development. Meanwhile, ML introduces a layer of predictive and adaptive capabilities, essential for navigating vast data landscapes and offering insights beyond the reach of traditional methods. Legal reasoning, with its intricate balance of logic, ethics, and practicality, offers a rich domain for applying and testing these technologies.

The thesis is predominantly focused on argumentation from both theoretical and technological perspectives, with the main objectives of advancing our understanding of argumentation in AI and offering practical tools and frameworks for its application, particularly in the context of legal reasoning. Nonetheless, toward the end, a crucial segue is made to bridge the gap between traditional, symbolic AI methods and the newer, data-driven approaches of ML. This convergence points towards a future where the rigour and clarity of symbolic methods, and argumentation in particular, help in the development of ML systems, paving the way for more robust, versatile, and ethically sound AI systems in legal and other domains.

The journey starts in the domain of meta-argumentation, a relatively unexplored yet pivotal aspect of argumentation theory, warranting a deeper exploration due to its far-reaching implications in the entire artificial intelligence area— and legal reasoning in particular. At its core, meta-argumentation involves the analysis and construction of arguments about arguments—a higher-order level of reasoning that addresses the interaction, composition, and evaluation of arguments themselves. This facet of argumentation theory opens the door to a nuanced understanding of how arguments conflict, complement, or supersede one another in complex reasoning scenarios.

Historically, the field of argumentation has seen various frameworks developed to structure and analyse arguments, such as ASPIC$^+$, ABA, and DeLP. These frameworks provide structured approaches to constructing and evaluating arguments, with each bringing its unique strengths and methodologies to the table. Despite their individual merits, these approaches often regard rules, conflicts, and preferences as given, without delving deeply into the meta-level discourse about these elements. In real-world adversarial debates, however, the rules themselves, the nature of conflicts, and the basis of preferences are often the subjects of contention and debate. A meta-argumentative approach seeks to address these higher-level concerns by providing frameworks and tools to argue about the components of argumentation frameworks themselves. Such an approach is vital in domains where arguments are not only about reaching a conclusion but also about contesting the foundational elements that lead to those conclusions.

Despite its importance, meta-argumentation has not received widespread attention in academic discourse. This oversight presents a notable gap in the study and application of structured argumentation, particularly in complex, real-world scenarios where the underlying rules and preferences are as critical as the arguments they support. For instance, in a legal context, not only the arguments presented in a case matter, but also the interpretation and validity of the laws and precedents that underpin these arguments.

This thesis endeavours to fill this gap by focusing on structured meta-argumentation within the ASPIC$^+$ framework. The ASPIC$^+$ framework stands as a cornerstone in the domain of structured argumentation, providing a systematic approach to constructing and evaluating arguments based on a set of premises and inferential rules. This framework is known for its structured methodology, offering a clear and logical process for argument formulation and assessment. However, despite its strengths, ASPIC$^+$ exhibits certain limitations, particularly when it comes to reasoning about its own foundational elements. This involves, for example, debating the validity of a rule within a given argumentative context, assessing whether there is an inherent conflict between two propositions, or determining the relative strength or preference of competing arguments. We aim at enhancing the framework's ability to represent and reason about the fundamental elements of argumentation, such as the validity of rules, the nature of conflicts, and the basis of preferences. The thesis also provides a simple yet powerful extension for allowing to reason about a concept quite important in the legal area, i.e., Burden of Persuasion. The aspiration is to develop a comprehensive framework that encapsulates these elements, thereby elevating the discourse in structured argumentation and broadening its applicability and relevance in both academic and practical realms.

In enhancing the ASPIC$^+$ framework within the scope of this thesis, a fundamental decision is made to ensure compatibility with Dung's seminal abstract argumentation semantics. This strategic choice, far from being merely conservative, is rooted in several critical considerations that align with both the overarching goals of our research and the broader field of argumentation theory.

First and foremost, adhering to Dung's semantics ensures a continuity of theoretical foundations crucial for the integrity and coherence of argumentation studies. Dung's framework, established in 1995, has since become a universally-recognised base in argumentation theory. Its well-established semantics offers a robust and reliable foundation, upon which further advancements, such as those proposed in this thesis, can be built. This compatibility is not merely a matter of convenience; it is a strategic choice that allows the enhanced ASPIC+ framework to be readily integrated into and compared with the existing body of work in the field.

Secondly, Dung's semantics are valued for their flexibility, a characteristic crucial in a field as diverse as argumentation theory. The ability to apply concepts across various domains and scenarios is a testament to the versatility of Dung's framework. By maintaining compatibility with these semantics, our enhanced framework inherits this adaptability, allowing it to be effectively employed in meta-argumentative contexts. This flexibility opens the door to multifaceted applications without necessitating extensive modifications or adaptations.

The decision to maintain compatibility with Dung's semantics also has practical and academic implications. In the academic realm, it ensures that our contributions are grounded in a widely accepted paradigm, facilitating broader acceptance and application. From a practical standpoint, the flexibility afforded by Dung's semantics means our framework can be applied to a range of real-world scenarios, enhancing its relevance and utility. The overall goal is to ensure that the advancements made in this thesis are not only theoretically sound and innovative but also practically applicable and relevant within the established traditions and future directions of argumentation theory. This approach exemplifies a balanced pursuit of innovation while respecting and building upon the foundational work that has shaped the field.

Until now, the discussion has primarily concentrated on theoretical advancements. However, it is fundamental not to overlook the technological perspective that plays a crucial role in this discourse. Thomas Edison famously once stated, "Vision without execution is just hallucination". Applying this perspective to the technological advancements in the Argumentation field reveals a somewhat disheartening reality. Despite the numerous and varied formal contributions made over the years, the resulting technologies have often struggled to match the rapid pace of innovation in this area. While the term "hallucination" might seem overly harsh, it is fair to say that the concept of a logic-based, technically mature environment for argumentation in intelligent systems has, until now, largely remained an elusive dream.

To balance this focus, there has been a marked focus on developing a stable and user-friendly framework capable of integrating all the formal contributions amassed during the course of this doctoral research. This emphasis aims at transforming the vision of advanced argumentation technologies into a tangible and practical reality, bridging the gap between theoretical innovation and functional application. By presenting the Arg2P framework, the thesis addresses the gap identified in the technological landscape of argumentation technologies. Designed to be neutral with respect to the underlying argumentation theory, Arg2P offers a versatile and flexible environment for integrating new theoretical contributions. The framework is tailored to fit advanced application scenarios in intelligent systems, including pervasive intelligent systems and the Internet of Things (IoT). It embodies key features of micro-intelligence, such as customisation of inference methods, situational awareness, and the capability to operate effectively at the micro-level across various contexts and architectures, and it is developed in line with current software engineering standards. Its modular architecture promotes system openness and ease of extension, aligning with the requirements of continuous integration and continuous delivery (CI/CD) practices. This design approach ensures that Arg2P is not just technically robust but also adaptable and scalable, meeting the evolving demands of modern intelligent systems.

There have been many issues to consider while trying to bridge the gap between the formal theoretical results in argumentation theory and their practical applications. Indeed, Argumentation, while rich in academic exploration and theoretical complexity, often grapples with practical implementation challenges, particularly in the integration of formal systems into everyday technological applications. The intricate nature of formal argumentation, with its web of premises, conclusions, and rebuttals, poses significant computational challenges. These challenges are heightened by the need to integrate these formal systems into contemporary technological environments, which are diverse and ubiquitous. The primary challenge, therefore, is to develop a compromise that allows the rigorous results of argumentation theory to be effectively and efficiently applied in practical scenarios. To address this, the thesis explores the modification of argumentation mechanisms to meet the requirements of real-world applications. This includes examining how the resolution process can be distributed and potentially simplified without compromising the theoretical integrity of the arguments. A key focus is on developing algorithms that are both theoretically sound and practically viable, capable of operating within the constraints of real-world computational environments.

As we transition from the more technical aspects of this thesis to its application in legal reasoning, it becomes evident that the innovations and advancements made in structured meta-argumentation and the Arg2P framework have significant implications in the legal domain. The journey from theoretical constructs and algorithmic developments to practical implementations is crucial, especially when considering that legal reasoning stands as one of the primary goals of this research. The intricate balance between theoretical rigour and practical applicability is of paramount importance in the legal field, where the precision and clarity of argumentation models directly impact the effectiveness and fairness of legal outcomes.

The application of the presented argumentation techniques in legal AI is not merely an extension of their technical capabilities but a vital test of their relevance and efficacy in real-world scenarios. Legal reasoning requires a framework that not only supports complex argumentative structures but also aligns with the nuanced and often intricate nature of legal discourse. The enhancements made in the ASPIC$^+$ framework, particularly in areas such as the burden of persuasion and conflict resolution, along with the adaptability and robustness of the Arg2P technology, offer new avenues for legal analysis and decision-making. These advancements provide the tools necessary for addressing the unique challenges of legal argumentation, including the interpretation of laws, evaluation of evidence, and resolution of disputes.

In essence, the transition to the application of these technologies in legal AI symbolises a crucial phase in this thesis. It underscores the practical significance of the research, demonstrating how theoretical models and technological innovations can converge to enhance the field of legal reasoning. Indeed, the formal advancements in the meta-argumentation model have shown a profound connection with legal reasoning. Chapter 6 illustrates how the meta-model facilitates nuanced analysis and interpretation of legal texts, especially focusing on legal interpretation. By applying the meta-model to interpretive challenges commonly faced in legal practice, such as conflicting precedents or ambiguous statutory language, the model demonstrates its capability to provide clarity and structure in complex interpretive tasks.

Following the focus on legal interpretation, the thesis shifts to illustrate the integration of our Arg2P technology into legal reasoning processes, showcasing the adaptability of our approach in handling the dynamic and multifaceted nature of legal problems. Through examples and case studies, the thesis demonstrates how Arg2P technology can help analyse representing and dealing with legal knowledge, enhancing the depth and clarity of traditional legal analysis and offering innovative solutions to longstanding legal challenges.

As we progress into the final chapters of the thesis, a pivotal shift in focus emerges, transitioning from the realms of pure symbolic AI to the burgeoning field of machine learning. This shift is reflective of a broader trend in the AI landscape, where the emphasis is increasingly on the integration and application of ML techniques. While symbolic AI, with its roots in logic, has laid a solid foundation, the advent of ML brings forth new paradigms, challenges, and opportunities, particularly in domains as critical as legal AI. Indeed, the evolution from symbolic AI to ML signifies a significant departure in approach and methodology. Symbolic AI, characterised by rule-based systems and logical reasoning, excels in scenarios where clear, well-defined rules and structured data prevail. However, the emergence of ML has opened new avenues for handling complex, unstructured data, learning from patterns, and making predictions, areas where traditional symbolic methods may falter. This transition is not just a technical progression but also a conceptual shift, demanding a reevaluation of existing frameworks and the development of new paradigms that can effectively harness the power of ML while addressing its inherent challenges.

In the legal domain, this shift is particularly consequential. The application of ML in legal settings offers promising avenues for predictive analysis, case outcome forecasting, and processing vast amounts of legal data. However, it also raises critical ethical considerations, such as bias, transparency, and accountability. These ethical concerns are not just peripheral issues but are central to the integrity and fairness of legal AI systems. Dealing with these issues involves documenting how data is sourced, how models are selected and trained, and how decisions are made. Such transparency is particularly vital in the legal field, where the reasons behind a decision can be as important as the decision itself. On top of that, these exact concerns have been raised also at a regulatory level with the European Union's AI Act, which stresses the safety, transparency, and accountability of all AI systems.

In the depicted context, automated machine learning (AutoML) has been introduced to automate the end-to-end process of applying machine learning to real-world problems. In a typical machine learning application, several steps are involved, including data preprocessing, model selection, and hyperparameter tuning. These steps are often complex and require significant domain expertise, making them extremely challenging. AutoML aims at simplifying these processes, making machine learning more accessible and efficient. Chapter 7 proposes a new approach to AutoML based on Structured Argumentation to provide Data Scientists with a symbolic interface for incorporating their expertise and navigate the development of ML systems in a manner that is both transparent and under control. By weaving argumentation techniques into the fabric of ML systems, developers and users are equipped with deeper insights into the reasoning processes of these systems. This integration ensures that the decision-making mechanisms of ML systems adhere to developers' desiderata, thus leading to an enhanced reliability and trustworthiness of the produced ML applications. The model is implemented by exploiting Arg2P as the core argumentation technology, thereby offering a tangible demonstration not just of the versatility and effectiveness of this innovative argumentation technology, but also of the vast potential inherent in the integration of argumentation and ML.

In conclusion, this thesis provides several significant contributions to the fields of argumentation theory and legal AI, articulated across its chapters. Chapter 3 focuses on enhancing the ASPIC$^+$ framework, introducing structured meta-argumentation to address its limitations in reasoning about rules, conflicts, and preferences. This is further enriched by integrating the concept of the burden of persuasion in the model, thereby expanding the framework's applicability in legal reasoning. Chapter 4 transitions into the practical applications of these theoretical advancements by introducing novel argumentation mechanisms to improve efficiency in real-world environments, and Chapter 5 showcases the development of a new argumentation technology, the Arg2P framework. Chapter 6 demonstrates the practical feasibility and adaptability of the newly introduced formal models and technology in the legal AI area. Finally, Chapter 7 moves the shift towards machine learning and proposes the use of structured argumentation as a tool to help AutoML development, highlighting its potential to bring transparency and control to ML processes.

Chapter 2

Background

This thesis is anchored on the exploration of two intertwined domains: formal argumentation and the technological reifications of formal argumentation models. The background chapter is aimed at laying a solid foundation in these areas, providing the reader with the fundamental notions essential for a comprehensive understanding of these fields. Each section within this chapter is dedicated to one of these subjects.

The first section delves into the domain of formal argumentation. Here, we give the fundamental notions of both abstract and structured argumentation frameworks, with a special focus on the ASPIC$^+$ framework and on meta-argumentation models. The section aims to elucidate the principles and mechanisms underlying formal argumentation, setting the stage for subsequent discussions on their practical applications and technological embodiments.

In the second section, we transition from theoretical frameworks to the technological reifications of formal argumentation models. This part of the chapter focuses on how theoretical constructs in argumentation have been concretized and implemented in various technologies.

The final section introduces the concept of automated machine learning (AutoML). This segment serves as a primer to AutoML, outlining its fundamental principles, methodologies, and its role in the field of machine learning. The inclusion of AutoML in this background chapter is strategic, as it prepares the ground for its significant involvement in later chapters, especially in relation to structured argumentation.

2.1 Argumentation

Logic is defined as the abstract study of statements, sentences and deductive arguments [Paulson, 2018]. From its birth, it has been developed and improved widely, now including a variety of formalisms and technologies.

Argumentation is a well-known formal tool for handling conflicting information (e.g., opinions and empirical data). In abstract argumentation [Dung, 1995a], a scenario (e.g., a legal case) can be represented by a directed graph. Each node represents an argument, and each edge denotes an attack by one argument on another. Each argument is regarded as atomic. There is no internal structure to an argument. Also, there is no specification of what is an argument or an attack. A graph can then be analyzed to determine which arguments are acceptable according to some general criteria (i.e., semantics) [Baroni et al., 2011a].

Let us start by defining a generic argumentation framework. This introduction has two purposes: *(i)* to give the reader with no knowledge in the formal argumentation field an idea of its main concepts and notions, (ii) to serve as a basis for the analysis contained in next chapters. For a more complete introduction we invite the reader to consult the vast amount of available literature on the argument [Baroni et al., 2018, Besnard et al., 2014b].

We first introduce the notion of *argumentation framework* and semantics according to Dung's original work [Dung, 1995b].

Definition 1 (Argumentation framework)**.** *An argumentation framework AF is a tuple* $< A, \rightsquigarrow >$*, where* A *is a set of arguments and* $\rightsquigarrow$ *is a binary relation (attack relation) over* $A \times A$*. We write* $X \rightsquigarrow Y$ *for* $(X, Y) \in \rightsquigarrow$*.*

The semantics – i.e., the way we use to give *meaning* to a framework – for an argumentation framework is defined as follows .

Definition 2 (Semantics)**.** *Let* $< \mathcal{A}, \rightsquigarrow >$ *be an AF and* $S \subseteq \mathcal{A}$*.* S *is conflict free iff there are no* $A, B \in S$ *such that* $A \rightsquigarrow B$*. For any* $X \in \mathcal{A}$*,* X *is acceptable with respect to* $S \subseteq \mathcal{A}$ *iff* $\forall Y \in \mathcal{A}$*,* $Y \rightsquigarrow X$ *implies that* $\exists Z \in S$ *s.t.* $Z \rightsquigarrow Y$*. Then:*

- *S is an admissible extension iff $X \in S$ implies that X is acceptable w.r.t. S;*

- *S is a complete extension iff $X \in S$ whenever X is acceptable w.r.t. S;*

- *S is the grounded extension iff S is the set-inclusion minimal complete extension;*

- *S is a stable extension iff attacks every argument that does not belong in it;*

A labelling-based definition of original Dung's semantics has also been provided [Baroni et al., 2011b]. Accordingly, each argument is associated with one label which is either IN, OUT, or UND—respectively meaning that the argument is either accepted, rejected, or undecided. Given a labelling for a framework, a IN, OUT, UND labelling for the statements claimed by the arguments in the graph can be also derived.

A way to link abstract argumentation and logical formalisms has been advanced in the field of structured argumentation [Besnard et al., 2014b], where we assume a formal logical language for representing knowledge (i.e., a LogicalKB) and for specifying how arguments and conflicts (i.e., attacks) can be derived from that knowledge. In the structured approach, the premises and claims of the argument are made explicit, and the relationship between them is formally defined through rules internal to the formalism. We can build the notion of attack as a binary relation over structured arguments that denotes when one argument is in conflict with another (e.g., contradictory claims or premises). One of the main frameworks for structured argumentation is ASPIC$^+$[Modgil and Prakken, 2014a]. In this formalism, arguments are built with two kinds of inference rules: strict rules, whose premises guarantee their conclusion, and defeasible rules, whose premises only create a presumption in favor of their conclusion. Then conflicts between arguments can arise from both inconsistencies in the LogicalKB and the defeasibility of the reasoning steps in an argument (i.e., a defeasible rule used in reaching a certain conclusion from a set of premises can also be attacked).

Let us introduce the main definitions of the ASPIC$^+$ framework.

We start with the *Argumentation system*, i.e., the structure containing all the elements that give to a derived framework its shape.

Definition 3 (Argumentation system). *An argumentation system is a quadruple AS=$< L, R, n, \rhd >$ where:*

- *L is a logical language;*
- *$R = R_s \cup R_d$ is a set of rules. R_d is a set of defeasible rules in the form $\phi_0, ..., \phi_n \Rightarrow \phi$, R_s is a set of strict rules in the form $\phi_0, \ldots, \phi_n \rightarrow \phi$, where $\phi_0, \ldots, \phi_n, \phi$ are well-formed formulae in the L language;*
- *n is a naming function of the form $n : R \mapsto L$*
- *$\rhd$ is a non-symmetrical conflict relation over $L \times L$. We write $\phi \rhd \psi$ for $(\phi, \psi) \in \rhd$.*

Then we have a *Knowledge base* containing both the axioms and the defeasible premises at the base of every argument built in the structured framework.

Definition 4 (Knowledge base). *A knowledge base for an AS=$< L, R, n, \rhd >$ is a set $K \subseteq L$ consisting of two disjoint subsets K_s (the axioms) and K_p (the ordinary premises).*

The union of an *Argumentation System* and a *Knowledge Base* produces an *Argumentation Theory*.

Definition 5 (Argumentation theory). *An argumentation theory is a tuple AT=$< AS, K >$ where AS is an argumentation system and K is a knowledge base in AS.*

Given an argumentation theory, by chaining rules from the theory we can construct arguments, as specified in the following definition; cf. [Caminada and Amgoud, 2007a, Modgil and Prakken, 2014b, Vreeswijk, 1997a].

Definition 6 (Argument). *Starting from an argumentation theory AT=$< AS, K >$, an argument A is any structure obtained by applying the following steps a finite number of times*

1. *ϕ if $\phi \in K$ with:*
   - *Prem(A)=$\{\phi\}$,*

- *Conc(A)=ϕ,*
- *Sub(A)=$\{\phi\}$,*
- *DefRules(A)=$\emptyset$,*
- *TopRule(A)=undefined*

2. $A_1, \ldots, A_n \Rightarrow \psi$ *if* $A_1, \ldots, A_n$ *are arguments s.t.* $\exists$ *a rule* $r = Conc(A_1), \ldots, Conc(A_n) \Rightarrow \psi \in R_d$*.*

- *Prem(A)=*$Prem(A_1) \cup \ldots \cup Prem(A_n)$*,*
- *Conc(A)=ψ,*
- *Sub(A)=*$Sub(A_1) \cup \ldots \cup Sub(A_n) \cup \{A\}$*,*
- *TopRule(A)=r,*
- *DefRules(A)=*$DefRules(A_1) \cup \ldots \cup DefRules(A_n) \cup \{r\}$

3. $A_1, \ldots, A_n \rightarrow \psi$ *if* $A_1, \ldots, A_n$ *are arguments s.t.* $\exists$ *a rule* $r = Conc(A_1), \ldots, Conc(A_n) \rightarrow \psi \in R_s$*.*

- *Prem(A)=*$Prem(A_1) \cup \ldots \cup Prem(A_n)$*,*
- *Conc(A)=ψ,*
- *Sub(A)=*$Sub(A_1) \cup \ldots \cup Sub(A_n) \cup \{A\}$*,*
- *TopRule(A)=r,*
- *DefRules(A)=*$DefRules(A_1) \cup \ldots \cup DefRules(A_n)$

Given an argument A we write:

- $Prem(A)$, for the set of ordinary premises from K_p used in the argument;
- $Conc(A)$, for the conclusion of the argument;
- $Sub(A)$, for the set of arguments supporting A's conclusion;
- $DefRules(A)$, for the set of rules in R_d used to build the argument;
- $TopRule(A)$, for the rule from R used in A's last inference step.

The first condition deals with arguments generated using the knowledge base K. Using the second and third ones we can recursively apply rules from R on the generated arguments to generate new arguments.

We can produce attacks starting from arguments using the notion of conflict for an argumentation language L:

Definition 7 (Direct attack). *An argument A directly attacks an argument B iff A directly undercuts, directly undermines or directly rebuts B where:*

- *A directly undercuts B iff $Conc(A) \rhd n(TopRule(B))$;*
- *A directly rebuts argument B iff $Conc(A) \rhd Conc(B)$ and $TopRule(B) \in R_d$;*
- *A directly undermines argument B iff $B \in K_p$ and $Conc(A) \rhd B$*

Definition 8 (Attack). *We say that argument A attacks argument B if A directly attacks $B' \in Sub(B)$.*

Then we can build an abstract argumentation framework as:

Definition 9 (Abstract argumentation framework). *Let AT be an argumentation theory $< AS, K >$. An abstract argumentation framework defined by AT, is a tuple $< \mathcal{A}, \rightsquigarrow >$ where:*

- *$\mathcal{A}$ is the set of all arguments constructed from AT according to Definition 6;*
- *for any arguments X and $Y \in \mathcal{A}$, $X \rightsquigarrow Y$ iff X attacks Y*

2.1.1 Meta-Argumentation

Modgil & Bench-Capon [Modgil and Bench-Capon, 2011] introduce the notion of meta-level argumentation frameworks. The arguments of meta-level argumentation frameworks make claims about object-level abstract argumentation frameworks according to the theory of such frameworks, for example, “A is in a preferred extension of AF” or “argument A in AF defeats argument B in AF”. Constraints are formulated on the attacks of the meta-level framework to ensure that such statements are correct with respect to the object level. For example, “y defeats x” attacks “x is justified”. This allows the formalisation of Dung’s theory of abstract argumentation frameworks in meta-level argumentation frameworks that have the same semantics as Dung's original frameworks. Moreover, Modgil & Bench-Capon show that the same approach can be used to formalise variants of Dung-style argumentation frameworks, such as preference- and value based AFs and extended AFs. In a similar way, Boella & al. [Boella et al., 2009b] develop a general methodology for instantiating Dung's original argumentation frameworks starting from extended argumentation frameworks through a flattening technique—comparably to what is done in [Gabbay, 2009]. The resulting framework operates on meta-arguments, for example in the form "argument A is accepted" while remaining in the formal framework of Dung's argumentation theory. While these approaches are theoretically very interesting, they do not specify the structure of arguments at the object level and therefore seem less suitable for knowledge representation.

Moving beyond abstract argumentation, [Olivieri et al., 2021] introduces a variant of defeasible logic, Defeasible Meta-Logic, to represent defeasible metatheories, by proposing algorithms to compute the (meta-)extensions of such theories, and by proving their computational complexity.

Wooldridge & al. [Wooldridge et al., 2005] develop a completely different approach for dealing with the meta-argumentative nature of argument systems. The work proposes a hierarchical first-order meta-logic, producing a three tiers argument system. Level 0 contains statements on the object domain, level 1 introduces the notion of arguments and acceptability, while level 2 is used to reason on the structure of arguments and their relations. This formalism – because of the required hierarchical representation –, although enabling a clear separation between meta- and object- level concepts, could result in decreased flexibility in the formalisation of the knowledge in the system.

A similar line of work based on the idea of using a multi-level argumentation system is available in the work from Müller & al. [Müller et al., 2013, Ogunniye et al., 2018]. In particular, the first formalisation of meta-argumentation synthesising bimodal graphs, structured argumentation, and argument schemes in a unique framework is presented in [Müller et al., 2013]. There, a formal definition of the meta-ASPIC framework is provided as a model for representing object arguments.

We now recall the main definitions of bimodal graphs as introduced by Müller

& al. [Müller et al., 2013] as they will be used in the next chapters. Bimodal graphs make it possible to capture scenarios where arguments are categorised in multiple levels—two in our case, the object and the meta level. Accordingly, a bimodal graph is composed of two components: an argumentation graph for the meta level and an argumentation graph for the object level, along with a relation of support that originates from the meta level and targets attacks and arguments on the object level. Every object-level argument and every object-level attack is supported by at least one meta-level argument. Meta-level arguments can only attack meta-level arguments, and object-level arguments can only attack object-level arguments.

Definition 10 (Bimodal argumentation graph)**.** *A* ***bimodal argumentation graph*** *is a tuple* $\langle \mathcal{A}_O, \mathcal{A}_M, \mathcal{R}_O, \mathcal{R}_M, \mathcal{S}_A, \mathcal{S}_R \rangle$ *where*

1. $\mathcal{A}_O$ *is the set of object-level arguments*

2. $\mathcal{A}_M$ *is the set of meta-level arguments*

3. $\mathcal{R}_O \subseteq \mathcal{A}_O \times \mathcal{A}_O$ *represents the set of object-level attacks*

4. $\mathcal{R}_M \subseteq \mathcal{A}_M \times \mathcal{A}_M$ *represents the set of meta-level attacks*

5. $\mathcal{S}_A \subseteq \mathcal{A}_M \times \mathcal{A}_O$ *represents the set of supports from meta-level arguments into object-level arguments*

6. $\mathcal{S}_R \subseteq \mathcal{A}_M \times \mathcal{R}_O$ *represents the set of supports from meta-level arguments into object-level attacks*

7. $\mathcal{A}_O \cap \mathcal{A}_M = \emptyset$

8. $\forall A \in \mathcal{A}_O \ \exists \ B \in \mathcal{A}_M : (B, A) \in \mathcal{S}_A$

9. $\forall R \in \mathcal{R}_O \ \exists \ B \in A_M : (B, R) \in \mathcal{S}_R$

The object-level argument graph is represented by the couple $(\mathcal{A}_O, \mathcal{R}_O)$, while the meta-level argument graph is represented by the couple $(\mathcal{A}_M, \mathcal{R}_M)$. The two distinct components are connected by the support relations represented by $\mathcal{S}_A$ and $\mathcal{S}_R$. These supports are the only structural interaction between the meta and the object levels. Condition (8) above ensures that every object-level argument is supported by at least one meta-level argument, whereas condition (9) ensures that every object-level attack is supported by at least one meta-level argument.

Perspectives of the object-level graph can be defined as:

Definition 11 (Perspective)**.** *Let* $G = \langle \mathcal{A}_O, \mathcal{A}_M, \mathcal{R}_O, \mathcal{R}_M, \mathcal{S}_A, \mathcal{S}_R \rangle$ *be a bimodal argumentation graph and let* L_S *be a labelling semantics. A tuple* $\langle \mathcal{A}'_O, \mathcal{R}'_O \rangle$ *is an* L_S*-perspective of* G *if* $\exists\ l \in L_S(\langle \mathcal{A}_M, \mathcal{R}_M \rangle)$ *such that*

- $\mathcal{A}'_O = \{\ A | \exists B \in \mathcal{A}_M \ \ s.t.\ l(B) = \mathsf{IN}, (B, A) \in \mathcal{S}_A \}$
- $\mathcal{R}'_O = \{\ R | \exists B \in \mathcal{A}_M \ \ s.t.\ l(B) = \mathsf{IN}, (B, R) \in \mathcal{S}_R \}$

Consequently, an object argument may occur in one perspective and not in another according to the results yielded by the meta-level argumentation graph. Under this setting, the role of conditions (8) and (9) becomes clear: every element in a lower level must be relevant w.r.t. the meta-level argumentation process—i.e. we can not have arguments that in no case can be part of a perspective.

Along the same line, bimodal graphs are exploited in [Ogunniye et al., 2018] for dealing with arguments sources' trust. In [Ogunniye et al., 2018] ASPIC+ is used instead of meta-ASPIC at the object level and on a set of meta-predicates related to the object level arguments and the schemes in the meta level. Both [Müller et al., 2013] and [Ogunniye et al., 2018] use critical questions as the source of attacks at the meta level.

An interesting connection could be drowned with the multi-sorted argumentation networks proposed in [Rienstra et al., 2011], and their reification in the modal fibring approach from [Barringer et al., 2012]. The main idea of their work is to allow different parts of a framework – called cells – to be evaluated under different semantics. In a nutshell, a set of arguments is a multi-sorted extension only if it is the union of the extensions computed on the qualified arguments – i.e., argument not defeated and defended from attacks coming from other cells – of the single cells composing the framework. The modal fibring approach from [Barringer et al., 2012] allows every cell to be represented as a separate argumentation framework, with possibility modality used to express inter-cell attacks within these frameworks. Their work could appear similar to the bimodal approach in the way different graphs are used to derive the final results, but there is an important difference to consider: the nature of the relation used to connect the different graphs. Bimodal graphs exploit a support relation to model the dependency of an N-level argument on an N+1-level argument, while multi-sorted networks are based on inter-cell attacks.

In the context of meta-argumentation, there is long line of work dealing with argumentation frameworks that allow for arguments about preferences. In [Prakken and Sartor, 1996] conflicts between mutually rebutting arguments are decided by preferences, which are established by arguments included in the same argumentation framework. A fix-point semantics is used to compute extensions including preference arguments.

Reasoning about preferences has been recently modelled by introducing a preference-based attack against attacks [Modgil and Prakken, 2010]. Dung & al. [Dung et al., 2019] expands this idea, by having a framework that includes attack arguments, as well as preference attack arguments against attacks. In this way, the framework obtained can be evaluated by using standard Dung semantics.

Despite all this interesting work, a widely acknowledged meta-argumentation framework has not yet been defined.

2.2 Argumentation Technologies

In the ever-evolving landscape of technology and innovation, a crucial transition occurs when we move from the realm of theoretical argumentation to the tangible realm of technological reifications. This section aims to bridge the gap between abstract ideas and their practical manifestations in the world of technology.

Theoretical argumentation lays the groundwork, providing a solid foundation of concepts, theories, and principles, offering a lens through which we can understand potential technological advancements, foresee challenges, and imagine future possibilities.

However, the true impact of these ideas is only felt when we step into the world of technological reifications. This is where theory meets practice, where ideas are transformed into concrete applications that shape our daily lives. Technological reifications embody the practical implementation of theoretical concepts, turning abstract notions into real-world solutions and innovations.

In this section, we will explore how theoretical concepts have been reified in various technological advancements.

2.2.1 Abstract argumentation

The recent landscape of computational argumentation dates back to Dung's foundational work on *abstract argumentation theory* [Dung, 1995a]. Most of the research efforts in the area have dwelled in the development of *technologies* based on the original Dung's work. Among the others, notable examples are ArguLab [Podlaszewski et al., 2011] – exposing a direct implementation of labelling algorithms and aimed at the application in the context of MAS –, ASPARTIX [Dvorák et al., 2020] – based on an answer set programming (ASP) encoding of the argumentation problem –, ConArg [Bistarelli and Santini, 2011] – a constraint-based framework –, and μ-toksia [Niskanen and Järvisalo, 2020]—based on SAT-solver and winner of the last ICCMA competition. Generally speaking, this strand of works focuses on the issues of complexity and efficiency intrinsic to argumentation theory. Recently, the International Competition on Computational Models of Argumentation (ICCMA)[1] has contributed to increase the general interest in those sorts of problems.

However, the abstract argumentation perspective is strongly biased towards a view wherein the overall aim of argumentation is about deciding the status of some claim and providing a justification for it, where the nature of "justification" is often tailored to some logical reasoning process. Generally speaking, argumentation is seen there as a somewhat one-sided process in which a single party merely presents a reasoned justification to a given claim. In many applications scenarios this can be enough, as for instance in decision-support or explanation-driven systems [Bench-Capon et al., 1993]. However, a generally-acknowledged objection to these sorts of approaches is that they totally fail to embrace the *dialectical nature of argumentation* as a full-fledged discourse and debate, which mostly fits real-world application scenarios—such as the legal ones. There, argumentation is seldom a matter of a single party defending a claim: instead, it is usually an

[1] `http://argumentationcompetition.org`

informed exchange of ideas and positions involving many different contributors. It is then perhaps surprising that the significant computational exploitation of well-established models for dialogue within philosophical, rhetorical, and linguistic analyses is just a relatively-recent phenomenon.

Although originally explored to a limited extent as a means for interacting with expert systems, the significant factor motivating nowadays the computational use of dialogue methods in argumentation can be found in supporting MAS applications, where the structured argumentation approach [Besnard et al., 2014c] seems to be most appropriate [Kok et al., 2012]. Indeed, abstract argumentation provides for a formal model that can hardly be separated from the data structures – that is, their representation and their interaction with the surrounding context – in real contexts. In fact, the application scenario heavily affects the dialogue and its outcome. Abstract models – and related tools – remain however interesting from the foundational perspective: efficient solvers can be exploited and integrated into structured contexts where the dialogue-based interaction of the argumentative process plays a more visible role.

2.2.2 Structured argumentation

In the field of structured argumentation, technological developments have not grown as fast as theoretical ones. Different models and approaches can be found, and a standard has not emerged yet. Moreover, technology reification is often neither up-to-date nor easily reachable. In general, works in this area can be categorised based on two main features: *(i)* their operation – namely, Dung's reduction or structural reasoning – and *(ii)* their reference model—namely, DeLP [García and Simari, 2004], Carneades [Gordon et al., 2007], ABA [Toni, 2014], and ASPIC [Modgil and Prakken, 2014a]. Dung's reduction labels those systems performing a mapping from the structured knowledge to an abstract framework in order to decide the admissibility of the arguments. On the other hand, structural reasoning tools do not exploit Dung's model: instead, they implement other algorithms to set the state of the arguments. In the following we briefly highlight the main features of each category.

DeLP. DeLP is the oldest one, and comes with a reference implementation[2], also available on the Tweety library.[3] However, the tool is quite aged and lays unmaintained on the website. The DeLP computational model is inherently structural, but contains many limitations in terms of abstraction, in particular when compared to the other models. Other interesting extensions have been proposed in the years [Alsinet et al., 2010, Alsinet et al., 2012], but no mature implementation has emerged.

Carneades. The Carneades technology exposes an implementation of its model [Gordon and Walton, 2016] offering both Dung's abstract reasoning and structural reasoning.[4] From a technological point of view, it is one of the best solutions that can be found. Written in Go [5] and with a recent implementation (even though not recently updated), Carneades is distributed as a web application and allows evaluations both in terms of Dung's model and according to their structured evaluator. Carneades represents a very promising technology: yet, further efforts should be devoted to make it practical and effective within the aforementioned challenging AI context.

ABA. In the ABA category one can find both systems belonging to the structural reasoning strand – such as proxdd [Toni, 2013], abagraph [Craven and Toni, 2016] and grapharg [Craven et al., 2013][6] – and to the abstract reductionist strand— such as ABAplus[7] and [Lehtonen et al., 2017]. ABAplus [Bao et al., 2017] exploits ASPARTIX as its abstract solver, and offers a pure propositional language to encode the knowledge: it does not deal with preferences over rules, and it only supports preferences over assumptions. Structural reasoning tools leverage the *dispute derivations* algorithm, an efficient algorithm to avoid the construction of the entire argumentation graph in the evaluation of the acceptability of an argument. Overall, it represents a promising framework for AI applications, even

[2] `http://lidia.cs.uns.edu.ar/delp_client/`

[3] `http://tweetyproject.org/`

[4] `http://carneades.fokus.fraunhofer.de/carneades/`

[5] `http://golang.org/`

[6] `http://robertcraven.org/proarg/`

[7] `http://www-abaplus.doc.ic.ac.uk/`

though not reified in a ready-to-use technology for AI pervasive MAS—in short, the tool is just a prototype.

ASPIC. Finally, ASPIC is one of the most flexible frameworks – in terms of the abstraction it provides – in the structured argumentation area and for sure the most widely known: it allows for the representation of all the main argumentation abstractions and provides all the most common semantics for argument evaluation. A number of works demonstrate how others models can be reformulated as an ASPIC instantiation [Modgil and Prakken, 2014a, van Gijzel and Prakken, 2012, García et al., 2020]. The main implementations available (Toast) [Snaith and Reed, 2012] is based on an abstract reductionist approach exploiting Dung-O-Matic[8] as its base solver. There have been some attempts to perform structural reasoning in MAS exploiting ASPIC: among the others, notable examples are Argue `tu`Prolog [Bryant et al., 2006] and the ASPIC Argumentation Engine[9]; yet, the resulting technologies can be classified as just proofs of concept.

Overall, from a technological perspective, many improvements are required in order to make existing tools really usable and effective in a distributed environment, as well as properly documented and easily deployable. For these reasons, a new trend has recently emerged in the argumentation field, also in relation to explainability. For instance, [Lehtonen et al., 2020] discusses how a direct declarative approach based on ASP can be developed, whereas [Caminada and Uebis, 2020] – implementing [Caminada, 2015] – shows how argument-based entailment can be brought closer to human intuition, by proposing the use of formal discussion as a bridge technology.

2.3 Automated Machine Learning

Given a machine learning task to solve, the Data Scientist (DS) collects raw data in arbitrary formats, builds up knowledge on both the problem and the data, translates such knowledge into *constraints*, designs and trains a model, and finally

[8] `http://arg-tech.org/index.php/projects/dung-o-matic/`

[9] `http://webspace.science.uu.nl/~prakk101/aspic/`

deploys the solution. Such a solution consists of a *ML pipeline*: a sequence of *Data Pre-processing transformations* ending with an *ML task*. The DS instantiates the pipeline among a large set of *algorithms*, which, in turn, can potentially have many *hyperparameters*. The accuracy of the deployed solution depends on finding both the best algorithms along with their hyperparameters within an exponential search space.

Automated machine learning (AutoML) tools assist the DS in finding such an ML pipeline. They leverage state-of-the-art optimization approaches to smartly explore huge search spaces of solutions. AutoML has been demonstrated to provide accurate performance, even in a limited time/iteration budget. In the early days, only the optimization of the ML task was addressed (but no pre-preprocessing). Auto-Weka [Kotthoff et al., 2019] formalized the problem as "combined algorithm selection and hyperparameter optimization": various ML algorithms and hyperparameters are tested over a dataset to find the most performing configuration. Such optimization was successfully implemented by leveraging Bayesian optimization [Frazier, 2018], a sequential strategy for global optimization: until a limit (budget) of iterations or time is reached, an increasingly accurate model is built on top of the previously explored configurations.

Recently, AutoML is no longer limited to optimizing just the ML task, but it also includes Data Pre-processing [Giovanelli et al., 2021b, Quemy, 2019]. In doing so, Auto-sklearn [Feurer et al., 2019] fixes the arrangement of the transformations a priori, without considering that the most performing arrangement changes according to the problem and dataset at hand. However, considering several arrangements translates into larger search spaces that are not easy to explore.

Several improvements have been made to let AutoML tools explore as many configurations as possible. Multi-fidelity methods [Falkner et al., 2018] (i.e., the use of several partial estimations to boost the time-consuming evaluation process) have been exploited. Meta-learning leverages the previous performance of pipeline instances on a wide range of different datasets to provide several recommendations for the dataset at hand, such as promising pipeline instances (possibly acting as an alternative to Bayesian optimization) and search spaces producing good performance. Yet, meta-learning per se performs poorly, because it provides coarse-grained recommendations, while it is beneficial in warm-starting Bayesian optimization (i.e., the suggested pipeline instances are visited at the beginning to boost the convergence process).

2.3.1 Towards Human-centered AutoML Approaches

As of now, the DS role in AutoML is limited to choosing the dataset to analyze, the validation technique (e.g., cross validation, hold out), and the metric to optimize (e.g., accuracy, F1 score). AutoML researchers aim at making ML accessible to a wider audience; this has been addressed first by improving automation and now by improving transparency, which also enables human intervention when needed. Auto-Weka [Kotthoff et al., 2019] and Auto-Sklearn [Feurer et al., 2019] enables non-expert users to build ML models, but but it is difficult for them to understand the inner workings of these models. Indeed, as advocated in [Drozdal et al., 2020], DSs require to understand the process to trust the proposed solutions. This direction, named "Human-centered AutoML", is pursued by both researchers and companies.

As to research contributions, we found plenty of visualization wrappers. In [Drozdal et al., 2020], the authors raise the need of incorporating transparency into AutoML: after a session interview, they discover that – out of all their proposed features – model performance metrics and visualizations are the most important information to DSs when establishing their trust in the proposed solutions. ATMSeer [Wang et al., 2019b] provides different multi-granularity visualizations to enable users to monitor the AutoML process and analyze the searched models. PipelineProfiler [Ono et al., 2021] offers interactive visualizations of the AutoML outputs and enables the reproducibility of the results through a Jupiter notebook. Other contributions enhance current AutoML techniques towards easier human-interactions by: (i) supporting ethic and fair constraints in Bayesian Optimization through a mathematical encoding [Perrone et al., 2021, Yaghini et al., 2021]; (ii) simplifying the usage of AutoML with symbolic annotations [Peng et al., 2020] and declarative languages [Kraska et al., 2013]; (iii) supporting fast feed-backs from AutoML (i.e., runs that are less time-consuming) by leveraging well-known mechanisms of the DBMS (e.g., lineage optimization) [Vartak et al., 2015, Xin et al., 2018]. Recently, MILE [Lee and Macke, 2020] has proposed to perform AutoML analysis with an end-to-end framework that reflect a DBMS (i.e., a query language + a lineage optimization).

Companies like Google and IBM are the ones most engaged in boosting the involvement of the human in the loop. Google Vizer [Golovin et al., 2017] and Google Facets[10] are the two main visualization tools. The former reveals details of the different hyperparameters tried in the optimization [Golovin et al., 2017], and the latter focuses on analyzing the output and recognizes biased AI (e.g., ML models that discriminate on sensible attributes such as gender). As to IBM, AutoAI [Wang et al., 2020] and AutoDS [Wang et al., 2021b] are the tools developed within the MIT-IBM Watson AI Lab. Specifically, the former enables non-technical users to define and customize their business goals as constraints. The latter assists the DS team throughout the CRISP-DM process (e.g., in data collection and pipeline design [Muller et al., 2019, Wang et al., 2021b] and in the augmentation of the DS's knowledge about the dataset features [Drozdal et al., 2020]).

Overall, several studies have been made to understand the proper design of a Human-centered AutoML tool. In [Pfisterer et al., 2019], the authors overview the main AutoML issues; while in [Khuat et al., 2022] authors suggest improvements towards the Human-centered shift. In [Gil et al., 2019, Xin et al., 2021, Crisan and Fiore-Gartland, 2021], interviews with DSs are conducted to reveal their perception of AutoML as well as their needs and expectations in the next-generation tools. The main insight is that the future of data science work will be a collaboration between humans and AI systems, in which both automation and human expertise are indispensable [Wang et al., 2019a]. To this end, AutoML should focus on: simplicity, reproducibility, and reliability [Xin et al., 2021, Crisan and Fiore-Gartland, 2021].

[10] `https://pair-code.github.io/facets/`

Chapter 3

A Meta-Argumentation Framework

Meta-arguments support conclusions about other arguments, their interaction, their composition or their evaluation. For instance, a meta-argument may conclude that other arguments are in conflict or that one of them is preferred over the other, or it may provide new rules or facts that can be used in building arguments.

Meta-argumentation has received little attention thus far. As discussed in [Besnard et al., 2014a] there are various approaches to generate argumentation frameworks (AFs) in terms of accounts of the structure of arguments and their relations (e.g. ASPIC+, ABA, classical argumentation, DeLP). However, most of these approaches regard rule sets, specifications of conflicts and preferences as given. In the reality of adversarial debate, these things can also be argued about. Hence the importance of meta-argumentation.

In this pivotal chapter of the thesis, we undertake the endeavor to address this notable gap in the field of structured meta-argumentation. The focus is to develop and present a comprehensive framework capable of encapsulating and facilitating reasoning about the fundamental elements of the ASPIC$^+$ argumentation framework. Indeed, the ASPIC$^+$ framework provides a structured approach to constructing and evaluating arguments. However, there exists a lacuna in its ability to comprehensively represent and reason about certain critical elements. Our objective is to bridge this gap by expanding its definition in three key areas:

- reasoning about the validity of rules of an argumentation theory, namely, assessing whether a rule should be generally accepted as appropriate to the argumentation domain;
- reasoning about the conflict function of an argumentation theory, namely, assessing whether there is a conflict between two propositions in the argumentation language, i.e., whether the arguments concerning those propositions are incompatible so that accepting one of them entails rejecting the other;
- reasoning about the validity of preferences, namely, assessing whether a preference should be used while evaluating the relative strength of two competing arguments.

The result will be a framework that not only aligns with the theoretical underpinnings of the ASPIC model but also enhances its applicability and relevance in complex argumentation scenarios. This endeavor is not only a contribution to the academic discourse in the field but also a step towards practical applications in areas – e.g. legal reasoning – where advanced argumentation models are indispensable.

The key point of this entire work is whether this can be done while maintaining the compatibility with traditional argumentation methods and models—namely, Dung's semantics [Dung, 1995b]. Indeed, in the process of enhancing and expanding the ASPIC framework, a crucial decision was made to maintain compatibility with Dung's seminal abstract argumentation semantics, rather than constructing an entirely new, ad-hoc mechanism. This choice, while appearing conservative at first glance, is rooted in several compelling justifications that align with the overarching goals of our research and the broader field of argumentation theory.

Firstly, by adhering to Dung's semantics, we ensure a continuity of theoretical foundations that is vital for the integrity and coherence of argumentation studies. Dung's framework, with its well-established semantics, offers a solid and universally recognised basis upon which further developments can be built. This compatibility is not merely a matter of convenience but a strategic choice that allows our enhanced framework to be readily integrated and compared with existing models and applications in the field. It ensures that our contributions are grounded in a widely accepted paradigm, facilitating broader acceptance and application.

Secondly, Dung's semantics provides a high degree of flexibility, making it adaptable to a variety of contexts and scenarios. This versatility is crucial in argumentation theory, where the ability to apply concepts across diverse domains is highly valued. By maintaining compatibility with these semantics, our framework inherits this flexibility, allowing it to be effectively employed in different fields, ranging from artificial intelligence to legal reasoning and beyond. It opens the door to multifaceted applications without necessitating extensive modifications or adaptations. Moreover, by aligning with Dung's semantics, we lay a foundation that is conducive to comparative studies, allowing researchers to systematically evaluate and contrast our enhanced framework with existing models. This alignment not only fosters a deeper understanding of the nuances and implications of various approaches but also encourages an evolutionary growth in the field, where new models are built upon and compared with established ones, leading to cumulative progress.

In this work, a key initial step involves demonstrating the effectiveness of the model within the context of grounded semantics. This choice is not arbitrary; it is grounded in both practical and theoretical considerations. Grounded semantics is recognised for its computational efficiency, particularly in real-world scenarios, being the only argumentation semantics with polynomial complexity. This efficiency makes it an ideal candidate for proving the soundness and applicability of our model in computational settings.

Furthermore, by focusing exclusively on grounded semantics, we can more clearly illuminate the core concepts and mechanisms underlying our proposed model. This focus should allow the readers to delve deeply into the foundational ideas without the added complexity and distraction of other, more computationally intensive semantics. The simplicity and clarity provided by this approach facilitate a better understanding and appreciation of the model's essential principles.

Successfully proving the model's efficacy within the domain of grounded semantics is a significant step, providing a strong foundation for further exploration and development. This approach not only establishes the soundness of the model but also sets the stage for future investigations into its applicability across other semantics.

In last section of this chapter, we discuss the model of the burden of persuasion in structured argumentation [Calegari et al., 2021d, Calegari and Sartor, 2020b] under a meta-argumentative approach, which leads to *(i)* a clear separation of concerns in the model, *(ii)* a simpler and more efficient implementation of the corresponding argumentation tool, *(iii)* a natural model extension for reasoning over the burden of persuasion concepts.

The proposed meta-argumentation framework for the burden of persuasion includes three ingredients: *(i) object-level argumentation* – to create arguments from defeasible and strict rules –, *(ii) meta-level argumentation* – to create arguments dealing with abstractions related to the burden concept using argument schemes (or meta-level rules) –, and *(iii) bimodal graphs* to define interaction between the object level and the meta level—following the account in [Müller et al., 2013]. Also in this case, despite the introduction of the multi-layering mechanism, we adhere to a similar guiding principle: focusing on enhancing the structural aspects of the framework, specifically through the addition of a new meta-layer, rather than devising a new set of semantics as was done in the original work [Calegari et al., 2021d, Calegari and Sartor, 2020b]. This approach aligns with the overarching strategy of building upon established foundations to introduce innovative elements, rather than creating entirely new semantic systems.

3.1 Reasoning with Rules

In this section we introduce the first meta-level feature of our framework, namely, the possibility to argue about the validity of rules. In this regard, we introduce two levels of applicability of a rule: *i)* the general validity of a rule—i.e. if the rule should be generally accepted as appropriate to the argumentation domain (e.g., in legal reasoning). Arguing at this level we enable the evaluation of different interpretations of the same legal disposition so to determine the most suitable one; *ii)* the contingent applicability of a rule—i.e. if the rule should be applied to the specific problem, given its general validity. In ASPIC only the second level is covered, since the application of each rule r can be undercutted by arguments concluding for the negation $\neg n(r)$ of the rule's name $n(r)$. The first level is addressed only implicitly, i.e., by assuming that all rules included in the rule-base can be included in any argument. We will drop this assumption, requiring that rules are used only if their validity can be established, i.e, we require that any argument using a rule includes a subargument to the effect that the rule is valid. More exactly, an argument A using a rule named r must include a subargument for $\mathsf{usable}(r)$. It may be argued that this outcome could also be archived in a standard ASPIC framework by supplementing the body of each rule r with a predicate $\mathsf{usable}(r)$. However, we believe that taking into a account validity in the argumentation inference is preferable since it avoids redundancy in the representation of rules, and corresponds to the way in which legal provision are expressed (it is not the case that each legal provisions includes the proposition that the same provision is valid).

We shall rely for this purpose on the naming function ($n(r)$) provided by ASPIC. More exactly, to deal with the validity of rules we include in the argumentation language the predicate $\mathsf{usable}(x)$, with x a formula of the language; the naming function will be responsible for the connection of x with a rule in R.

Definition 12 (Rules Language)**.** *Given a logical language L we define a language for reasoning with rules L_r as the language $L_r = L \cup \{\mathsf{usable}(\psi)|\psi \in L\}$.*

All Definitions given in Section 2.1 remain valid. We just need to give a new *argument* definition – with respect to the one given in [Modgil and Prakken, 2014b] – enforcing the check on the validity of rules. In simple terms, every argument using a rule r, must include a subargument having $\mathsf{usable}(r)$ as conclusion (condition 2 and 3 of Definition 6).

Definition 13 (Arguments)**.** *Starting from an argumentation theory AT=$< AS, K >$, an argument A is any structure obtained by applying the following steps a finite number of times*

1. *ϕ if $\phi \in K$ with:*

   - *Prem(A)=$\{\phi\}$,*
   - *Conc(A)=ϕ,*
   - *Sub(A)=$\{\phi\}$,*
   - *TopRule(A)=undefined,*
   - *DefRules(A)=$\emptyset$*

- *LDefRules(A)*$=\emptyset$

2. $A_0, A_1, \ldots, A_n \Rightarrow \phi$ *if* $A_0, \ldots, A_n$ *are arguments s.t.* $\mathsf{Conc}(A_0) = \mathsf{usable}(n(r))$ *and* $\exists$ *a rule* $r = \mathsf{Conc}(A_1), \ldots, \mathsf{Conc}(A_n) \Rightarrow \phi \in R_d$.
   - *Prem(A)*$=\mathsf{Prem}(A_0) \cup \ldots \cup \mathsf{Prem}(A_n)$,
   - *Conc(A)*$=\phi$,
   - *Sub(A)*$=\mathsf{Sub}(A_0) \cup \ldots \cup \mathsf{Sub}(A_n) \cup \{A\}$,
   - *TopRule(A)*$=r$,
   - *DefRules(A)*$=\mathsf{DefRules}(A_0) \cup \ldots \cup \mathsf{DefRules}(A_n) \cup \{r\}$
   - *LDefRules(A)*$=\{r\}$

3. $A_0, A_1, \ldots, A_n \rightarrow \phi$ *if* $A_0, \ldots, A_n$ *are arguments s.t.* $\mathsf{Conc}(A_0) = \mathsf{usable}(n(r))$ *and* $\exists$ *a rule* $r = \mathsf{Conc}(A_1), \ldots, \mathsf{Conc}(A_n) \rightarrow \phi \in R_s$.
   - *Prem(A)*$=\mathsf{Prem}(A_0) \cup \ldots \cup \mathsf{Prem}(A_n)$,
   - *Conc(A)*$=\phi$,
   - *Sub(A)*$=\mathsf{Sub}(A_0) \cup \ldots \cup \mathsf{Sub}(A_n) \cup \{A\}$,
   - *TopRule(A)*$=r$,
   - *DefRules(A)*$=\mathsf{DefRules}(A_0) \cup \ldots \cup \mathsf{DefRules}(A_n)$
   - *LDefRules(A)*$=\mathsf{LDefRules}(A_0) \cup \ldots \cup \mathsf{LDefRules}(A_n)$

As usual, given an argument A we write:

- $\mathsf{Prem}(A)$, for the set of ordinary premises from K_p used in the argument;
- $\mathsf{Conc}(A)$, for the conclusion of the argument;
- $\mathsf{Sub}(A)$, for the set of arguments supporting A's conclusion;
- $\mathsf{DefRules}(A)$, for the set of rules in R_d used to build the argument;
- $\mathsf{TopRule}(A)$, for the rule from R used in A's last inference step;
- $\mathsf{LDefRules}(A)$, for the set of rules in R_d used in the last inference step,

Attacks are then defined in the standard way—i.e., Definition 7 and Definition 8.

The final Argumentation Framework is built using Definition 9 and evaluated with standard Dung's semantics (Definition 2). If we put all the $\mathsf{usable}(r)$ expressions where r is a rule identifier in K_s we should obtain a framework equivalent to standard ASPIC w.r.t. Dung's semantics.

Example 1 (Relation with standard ASPIC)**.** *In the following example, we assume that for every $\phi \in L$, $\phi \rhd \neg\phi$ and $\neg\phi \rhd \phi$. Let us consider the theory where $K_p = \{a, c\}$, $K_s = \{\mathsf{usable}(p), \mathsf{usable}(q)\}$ where $p : a \Rightarrow b$ and $q : c \Rightarrow \neg p$, and arguments:*

A0 : $\mathsf{usable}(p)$	*A3* : c
A1 : $\mathsf{usable}(q)$	*A4* : *A0*, *A2* $\Rightarrow b$
A2 : a	*A5* : *A1*, *A3* $\Rightarrow \neg p$

Since both rules p and q are strictly valid ($A0$ and $A1$), we can use them in combination with the ordinary premises a and c, to derive the statements b and $\neg p$. In this scenario the classical ASPIC+ rules apply: $A5$ delivers an undercutting attack against $A4$, hence b is rejected according to grounded (and all derived) semantics.

Note that the extension provided in this section does not represent a substantial departure from ASPIC. In fact, the same behaviour (the application of each rule being conditioned to its validity) could be obtained in the standard ASPIC framework by expanding the body of each rule r with the additional predicate $\mathsf{usable}(r)$.

3.2 Reasoning with Conflicts

According to Definition 3, the conflict relation is a fixed part of the argumentation system and attacks between arguments are determined by conflicts between the conclusion of the attacking argument and a premise, rule name, or conclusion of the directly attacked argument. The main idea underpinning the extension for dealing with meta-argumentation is to make the conflict relation dynamic, allowing arguments to argue for or against the existence of conflicts. In such a way we define an abstract argumentation framework that – once evaluated according to a standard Dung's semantics – produces admissible extensions containing both the arguments arguing on conflicts and arguments whose admissibility is influenced by these conflicts.

Let us start providing definitions for an argumentation language L enabling conflicts between elements of L to be stated, i.e., enabling reasoning with conflicts. For this purpose we need a language, which includes, for every couple of formulae ϕ and ψ, the predication of their contrariness, $\mathsf{contr}(\psi, \phi)$. More exactly, the predicate $\mathsf{contr}(\psi, \phi)$ expresses the idea that ψ is a *contrary* of ϕ, in the sense that the claim of ψ entails the rejection of ϕ.

Definition 14 (Conflict-based argumentation language). *Given an argumentation language L we define a argumentation language for reasoning with conflicts L_c as the smallest argumentation language $L_c = L \cup \{\mathsf{contr}(\psi, \phi) | \psi, \phi \in L_c\}$.*

Now let us consider L_c a language as in Definition 14. We can build an argumentation system AS=$< L_c, R, n, \emptyset >$ and consequently an argumentation theory AT=$< AS, K >$, and use them to build an abstract argumentation framework AF=$< \mathcal{A}, \rightsquigarrow >$ using Definition 9. Note that, since $\rhd = \emptyset$, the attack set $\rightsquigarrow$ in AF will be empty as well.

Now, let us extend the AF framework so defined to introduce attacks derived from the conflicts reified in the L_c language. In such a way the status of an attack is bound to the status of the argument claiming the conflict that generated it.

First, let us define an argument for each potential attack deriving from contr predicates. Accordingly, attacks could be evaluated w.r.t. the semantics applied to the framework.

Definition 15 (Conflict-based direct attack argument). *A conflict-based direct attack argument X stating that argument W, based on conflict argument W', attacks argument Z, has the form $W, W' \Rightarrow att(Z)$ where:*

- *$Conc(W) = \phi$, $Conc(W') = \mathsf{contr}(\phi, \psi)$ and*
  - *$n(TopRule(Z)) = \psi$ and $TopRule(Z) \in R_d$, or*

  - $Conc(Z) = \psi$ *and* $TopRule(Z) \in R_d$ *or*
  - $Conc(Z) = \psi$ *and* $Z \in K_p$

- $Conc(X) = att(Z)$
- $Sub(X) = Sub(W) \cup Sub(W') \cup \{X\}$

Let us write $DirectAttack(X)$ *to indicate that* X *is a direct attack argument.*

Thus to construct a *direct attack argument* $W, W' \Rightarrow att(Z)$ against Z it must be the case two arguments are available, argument W, and argument W', the latter claiming that the conclusion of W is in conflict with the relevant element of Z (i.e., the name of Z's top rule or Z's conclusion). The status of the direct attack arguments will depend on the status of both W and W'.

We leverage direct attack arguments to build the actual *attack* set of the meta argumentation framework.

Definition 16 (Conflict-based attack)**.** *A direct attack argument* $W, W' \Rightarrow att(Z)$ *attacks any argument* Z' *such that* $Z \in Sub(Z')$.

Thus, a direct attack argument $W, W' \Rightarrow att(Z)$ attacks not only its direct target Z, but also any argument Z' of which Z is a subargument. The success of the attack will depend not only on the status of W, but also on the status of W' which asserts that W and Z are in conflict.

These elements are merged together in a *Conflict-based Argumentation Framework.*

Definition 17 (Conflict-based Argumentation Framework)**.** *Given an argumentation theory AT=*$<< L_c, R, n, \emptyset >, K >$ *with* L_c *being a conflict-based argumentation language, the conflict-based argumentation framework of AT is the tuple* $< \mathcal{A}_1 \cup \mathcal{A}_2, \rightsquigarrow >$ *where:*

- $\mathcal{A}_1$ *is the set of all arguments constructed from AT according to Definition 6;*
- $\mathcal{A}_2$ *is the set of all direct attack arguments constructed from AT and* A_1 *according to Definition 15;*
- $X \rightsquigarrow Z$ *iff* X *attacks* Z *according to Definition 16*

Conflict freeness, acceptability, admissible, complete, grounded extension are defined as in Definition 2.

The set $\mathcal{A}_2$ contains all attack arguments that can be generated by using the arguments in A_1, according to Definition 17. For an attack argument $W, W' \Rightarrow att(Z)$ to be established according to an argumentation semantics, it is necessary that also W' is acceptable, i.e., that it is established that an acceptable conflict between W and Z exists. Only in this case W will bring an attack against Z.

We now proceed to demonstrate two important properties of the constructed framework. Intuitively, we would expect that a conflict that has been proven to exist at the meta-level – i.e. via the conflict-based framework –, indeed exists at the object level, leading to the same set of attacks and, consecutively, to the same extension. In other words, what is true according to the conflict-based framework should remain true when the verified conflicts are applied a priori as in the original ASPIC$^+$ model. To demonstrate this important property, let us introduce the notion of an *Equivalent Standard AF*.

To start, we define a way to construct a standard argumentation framework on the basis of a conflict-based argumentation framework. The basic idea is that starting with a conflict-based argumentation framework and an extension of it, we construct a standard argumentation framework having a corresponding extension according to the same semantics.

Let us consider a conflict-based argumentation framework CAF=$< \mathcal{A}, \rightsquigarrow >$ and one of its extensions E. To construct the equivalent argumentation framework EAF, we first remove from $\mathcal{A}$ (a) all attack arguments that are supported by those conflict arguments that are in the extension, and (b) all attack arguments that are supported by a conflicting argument that is attacked by the extension. Only attack arguments that are neither included in E nor attacked by it are left in the EAF's arguments set. Accordingly, the EAF's attack relation is constructed using those conflicts claimed by the arguments in E.

Definition 18 (Equivalent Standard AF). *Given a conflict based argumentation framework CAF=$< \mathcal{A}, \rightsquigarrow >$ having an extension E according to semantics σ, we define an equivalent standard argumentation framework EAF=$< \mathcal{A}', \rightsquigarrow' >$ where:*

1. $\mathcal{A}' = \mathcal{A} \setminus B \cup C$ *where:*

(a) *$B = \{a \in \mathcal{A} \mid \exists b \in E$ such that $Conc(b) = \mathsf{contr}(\phi, \psi)$ and a is a direct attack argument of the form $W, b \Rightarrow Z\}$;*

(b) *$C = \{a \in \mathcal{A} \mid \exists b \in \mathcal{A}$ such that $Conc(b) = \mathsf{contr}(\phi, \psi)$ and b is attacked by E and a is a direct attack argument of the form $W, b \Rightarrow Z\}$.*

2. *$\rightsquigarrow' = \rightsquigarrow_{|\mathcal{A}' \times \mathcal{A}'} \cup \{(a, b) \mid a, b \in \mathcal{A}'$ and $\exists c \in E$ such that $Conc(c) = \mathsf{contr}(\phi, \psi)$ and $\exists\ b' \in Sub(b)$ s.t. a directly attacks b' (Definition 7) according to the conflict $\phi \rhd \psi\}$.*

Proposition 1. *Consider a finitary CAF=$< \mathcal{A}, \rightsquigarrow>$ and its corresponding EAF=$< \mathcal{A}', \rightsquigarrow'>$ built on the grounded extension E and having grounded extension E'. Then $E \cap \mathcal{A}' = E'$.*

Proof 1. *Let's consider an argumentation framework $CAF = < \mathcal{A}, \rightsquigarrow>$. We call the characteristic function of CAF the function $F : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ such that $F(Args) = \{X | \forall Y$ such that $Y \rightsquigarrow X$, then $\exists Z \in Args$ such that $Z \rightsquigarrow Y\}$ where $Args \subseteq \mathcal{A}$. Let us consider grounded extension as the minimal conflict-free fixed point of the characteristic function F—i.e. the union of a sequence $E_0, \ldots, E_n$ obtained by iterative application of the F function on the empty set, and where $E_0 = \emptyset$. We prove that $E \cap \mathcal{A}' = E'$.*

We first prove that $E \cap \mathcal{A}' \subseteq E'$. Suppose $a \in E \cap \mathcal{A}'$. We prove that $a \in E'$ as follows.

Base case: *a has no attackers in $\mathcal{A}$ according to $\rightsquigarrow$ so $a \in E_1 \cap \mathcal{A}'$. Then there can only be attackers of a in $\mathcal{A}'$ according to $\rightsquigarrow'$ if there is a relevant conflict argument b in E that says that the conclusion of some argument $x \in \mathcal{A}$ conflicts with a's conclusion. But then there exists a direct attack argument $x, b \Rightarrow att(a)$ in $\mathcal{A}$, which contradicts that a has no attackers in $\mathcal{A}$. So $x \notin \mathcal{A}'$, so a has no attackers in $\mathcal{A}'$, so $a \in E'$.*

Induction step: *Assume that all arguments in E_{i-1} are in E'. Consider any $a \in E_i$. Any $b \in \mathcal{A}'$ such that $b \rightsquigarrow' a$ is such that $b \rightsquigarrow a$ or $b \not\rightsquigarrow a$. First, any such b such that $b \rightsquigarrow a$ is attacked by E_{i-1} according to $\rightsquigarrow$. Then by the induction hypothesis, if $b \in \mathcal{A}'$, then b is also attacked' by E. Next, consider any such b such that $b \not\rightsquigarrow a$. Then there is a direct attack argument $m \in \mathcal{A}$ of the form $b, X' \Rightarrow att(a)$. Then $m \rightsquigarrow a$ so there exists an $m' \in E_{i-1}$ such that $m' \rightsquigarrow m$.*

Note that m is a direct attack argument, so m is of the form $c, Z \Rightarrow att(b)$. By closure of E under subarguments (an easy adaptation of the same result on standard ASPIC+), c and Z are also in E_{i-1}. But then by the induction hypothesis $c \in E'$ and $c \rightsquigarrow' b$. So $a \in E'$.

We next prove that $E' \subseteq E \cap \mathcal{A}'$. Suppose $a \in E'$. We prove that $a \in E$ as follows.

Base case: *a has no attackers in $\mathcal{A}'$ so $a \in E'_1$. Consider any $b \in \mathcal{A}$ such that $b \rightsquigarrow a$. Then b is a direct attack argument of the form $c, X \Rightarrow att(a)$, with X a conflict argument that says that the conclusion of argument $c \in \mathcal{A}$ conflicts with a's conclusion. But since a has no attackers in $\mathcal{A}'$, we have that $b \notin \mathcal{A}'$ because of either condition (1a) or condition (1b) of Definition 18. In the case of (1b), b is attacked according to $\rightsquigarrow$ on X by an argument in E. In the case of (1a), we have $X \in E$, so, according to condition (2) of Definition 18, we have that $c \rightsquigarrow' a$. But this contradicts that a has no attackers in $\mathcal{A}'$. The two cases together prove that $a \in E$.*

Induction step: *Assume that all arguments in E'_{i-1} are in E. Consider any $a \in E'_i$. Then all $b \in \mathcal{A}'$ such that $b \rightsquigarrow' a$ are attacked by E'_{i-1} according to $\rightsquigarrow'$. Then b could be either a regular argument or a direct attack argument of the form $c, X \Rightarrow att(a)$. In the latter case, since $b \in \mathcal{A}$ then, by the induction hypothesis, b is also attacked by E according to $\rightsquigarrow$. In the first case, since $b \rightsquigarrow' a$, there must exist a direct attack argument $m \in \mathcal{A}$ of the form $b, X \Rightarrow att(a)$ such that $m \rightsquigarrow a$. But, by induction hypotheses, b and m are both attacked according to $\rightsquigarrow$ by E. The two cases together prove that $a \in E$.*

Since $E \cap \mathcal{A}' \subseteq E'$ and $E' \subseteq E \cap \mathcal{A}'$ then $E' = E \cap \mathcal{A}'$.

The second property we want to demonstrate builds on top of what we have just proven. We have seen that it is possible to move the conflicts in the grounded extension at the object level without altering the results. However, the resulting *Equivalent Standard AF* still contains the meta-level attack arguments that are not in the extension or attacked by a member of it. The question is whether there are cases in which the conflict framework can be completely transformed into a regular argumentation framework. The implication of this finding would be straightforward: the Conflict-based framework would be a generalisation of a regular abstract argumentation framework. This is a fundamental property for every model trying to provide a conservative extension like ours.

The next proposition shows that a *Conflict-based Argumentation Framework* is a generalisation of a standard abstract argumentation framework.

Proposition 2. *Consider a CAF=$< \mathcal{A}, \rightsquigarrow >$ and its corresponding equivalent EAF=$< \mathcal{A}', \rightsquigarrow' >$ built on the σ extension E. If $\forall x \in \{a \in \mathcal{A} | Conc(a) = \textsf{contr}(\phi, \psi)\}$ we have that either $x \in E$ or $\exists (d, x) \in \rightsquigarrow$ s.t. $d \in E$, then EAF is a regular argumentation framework as in Definition 9.*

Proof 2. *By Definition 18 if all the conflict arguments are either in the extension or attacked by a member of it, then all the Direct Attack Arguments can be discarded leaving only the arguments produced using Definition 6. The attack set would then be given by the set of conflicts claimed by the argument in the extension using Definition 8. Consequently, the result is a regular argumentation framework as in Definition 9.*

In the general case, however, we cannot have a complete equivalency between a CAF and a regular framework. Indeed, if an argument for $conf(\psi, \phi)$ is undecided – neither in the extension nor attacked by one of its members –, then the uncertainty can be propagated to the attack argument and then to the attacked argument, thus preventing them to be part of the extension. We could not obtain the same result without considering the Direct Attack Argument, because the absence of the conflict would potentially allow the attacked argument to be accepted without considering the potential uncertainty in the state of the conflict. In other words, a CAF framework is capable of conveying more information on the state of an attack w.r.t. a standard argumentation framework, thus making the transformation to a regular framework impossible in the general case.

Example 2 (Partial Transformation)**.** *Let us consider the theory where $K_p = \{p, q, r, -r\}$ and $K_s = \{conf(r, -r), conf(-r, r)\}$ and $R_d = \{r => conf(p, q)\}$. Starting from this theory we can build the Conflict-based framework and then the Equivalent one as shown in Figure 3.1.*

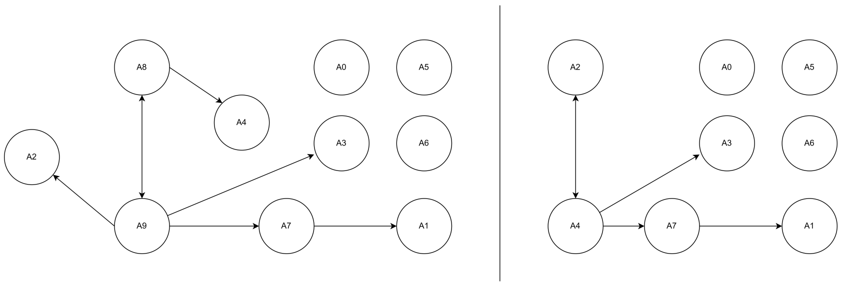
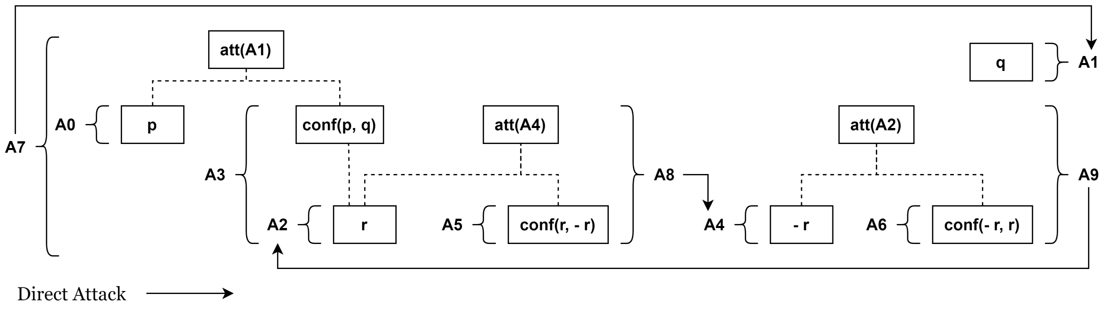


Figure 3.1: Conflict-based Argumentation framework from Example 2 on the left, Equivalent framework on the right.



If we apply Dung's grounded semantics to the frameworks, in both cases we obtain the extension $\{A0, A5, A6\}$. It can be noticed that the Equivalent framework still contains an attack argument ($A7$) due to the uncertainty in $A3$'s evaluation. Indeed, without knowing if $A3$ is in the extension or definitely rejected – i.e. attacked by a member of the extension –, it is impossible to decide whether $A0$ should attack $A1$ or not in the Equivalent Standard AF. Consequently, every alteration of the Equivalent attack set on the basis of this conflict would lead to a possible modification in the semantics results—i.e. the attack argument $A7$ with the connected attacks must be preserved in the Equivalent AF.

3.3 Reasoning with Preferences

The idea behind the extension for dealing with defeasible preferences is simple: for every argument involving a rule or a premise for which a preference is present in the framework – i.e. it is available an argument concluding the preference –, we create a new argument that takes that preference into account. In other words, given an argument, we will have in the framework one additional argument for every possible combination of its related preferences. For example, given an argument A built using the ordinary premise a and the rule r_0, and two arguments B and C concluding respectively $\mathsf{sup}(a, b)$ and $\mathsf{sup}(r_0, r_1)$, where $\mathsf{sup}(a, b)$ represents the superiority of a over b, and $\mathsf{sup}(r_0, r_1)$ the one of r_0 over r_1, then we can build three additional arguments:

- A_B, merging A with the preference claimed by B—$\mathsf{sup}(a, b)$;
- A_C, merging A with the preference claimed by C—$\mathsf{sup}(r_0, r_1)$;
- $A_{B \wedge C}$, merging A with the preferences claimed by both B and C

Let us now imagine a scenario where A attacks – and it is attacked – by a second argument $\overline{A}$: if we do not take any preference into account, the same should apply also to A_B, A_C and $A_{B \wedge C}$—they are built on A so attack relations are preserved. However, the idea of the extension is exactly to use the preferences on which the new arguments are based to determine if it is possible for $\overline{A}$ to defeat them. Let's suppose that A is preferred to $\overline{A}$ according to the preference $\mathsf{sup}(r_0, r_1)$—for sake of simplicity we are not considering the details of the ordering, we just know that without $\mathsf{sup}(r_0, r_1)$ the two arguments would have equal strength. Accordingly, we have that:

- $\overline{A}$ attacks and defeats A—no preferences are involved (this would not always be the case according to ASPIC+ orderings because also the structure of the arguments has to be considered);
- $\overline{A}$ attacks and defeats A_B—the preference $\mathsf{sup}(a, b)$ is not relevant;
- $\overline{A}$ attacks but does not defeat A_C—A is preferred to $\overline{A}$ according to the preference claimed by C;
- $\overline{A}$ attacks but does not defeat $A_{B \wedge C}$—A is preferred to $\overline{A}$ according to one of the preferences taken into account (of course there would be cases where more than a single preference is required to determine the relative strength of two arguments)

If we evaluate the framework with Dung's grounded semantics we obtain that A is included in the final extension and $\overline{A}$ is not. The reason is simple: $\overline{A}$ attacks and it is attacked by A and A_B, but, given the preferences, it can not return the attack from A_C and $A_{B\wedge C}$. Consequently, $\overline{A}$ is refuted and all the A-based arguments are allowed to be included in the extension. More importantly, the final result is subordinated to the inclusion of the arguments claiming the relevant preferences – C in the example – in the extension. Let us consider the case in which B is refuted. The extension would change but the result would not—i.e., also A_B and $A_{B\wedge C}$ would be refuted, but we would still have a valid defeat from A_C to $\overline{A}$, determining the exclusion of the latter. Conversely, in the case of C's rejection, both A and $\overline{A}$ would be excluded from the extension being A_C and $A_{B\wedge C}$ both refuted.

The work presented in [Dung et al., 2019] differs with the model presented in this section in two elements that we believe to be important especially, but not only, for legal reasoning. Firstly, Dung's model does not allow for the use of groups of preferences in the comparison of two arguments, thus excluding the use of most of the orderings introduced in ASPIC$^+$ ([Dung et al., 2019] is based on the use of a *normal* ordering only requiring a single preference). Secondly, albeit both our method and Dung & al. [Dung et al., 2019]'s are based on the introduction of additional arguments in the framework, we believe that our use of variations of original arguments – rather than introducing instantiations of attacks as additional arguments as in [Dung et al., 2019] – helps to maintain a higher degree of clarity and explicability—features for which standard argumentation frameworks are known for.

For this purpose we need first to define a preference language, which includes, for every couple of formulae ϕ and ψ, the predicate $\mathsf{sup}(\psi, \phi)$, expressing the superiority of ψ over ϕ (ϕ and ψ can represent rules through the naming function).

Definition 19 (Preference Language)**.** *Given an argumentation language L we define a preference language L_p as the smallest language $L_p = L \cup \{\mathsf{sup}(\psi, \phi) | \psi, \phi \in L\}$.*

We can then define Preference Arguments, i.e. arguments grouping a set of preferences related to a target argument:

Notation 1 (Defeasible Premises). *Given an argument A, we call $\mathsf{DefPrem}(A)$ the set $\mathsf{Prem}(A) \cap K_p$—i.e., the ordinary premises used to build the argument.*

Definition 20 (Preference Argument). *Let AT be an argumentation theory $< AS, K >$ based on a preference language, and $\mathcal{A}$ a set of arguments constructed from AT. Given $A \in \mathcal{A}$, the set of its defeasible elements $\mathsf{Def}(A)$ is $\mathsf{DefPrem}(A) \cup \{n(r) | r \in \mathsf{DefRules}(A)\}$. Then, the set of arguments claiming a preference supporting A is $S_A = \{B \in \mathcal{A} | \mathsf{Conc}(B) = \mathsf{sup}(\phi, \psi)$ and $\phi \in \mathsf{Def}(A)\}$. For any non empty $\{A_0, \ldots, A_m\} \subseteq S_A$, the corresponding preference argument B has the form:*

- $A_0, \ldots, A_m, A_{m+1} \ldots, A_n \Rightarrow \phi$ *if* $A = A_{m+1}, \ldots, A_n \Rightarrow \phi$,
- $A_0, \ldots, A_m, A_{m+1} \ldots, A_n \rightarrow \phi$ *if* $A = A_{m+1}, \ldots, A_n \rightarrow \phi$,
- $A_0, \ldots, A_m \Rightarrow \phi$ *if* $A = \phi$ *and* $\phi \in K_p$,
- $A_0, \ldots, A_m \rightarrow \phi$ *if* $A = \phi$ *and* $\phi \in K_s$,

where:

- $\mathsf{Prem}(B)$=$\mathsf{Prem}(A)$,
- $\mathsf{Conc}(B)$=$\mathsf{Conc}(A)$,
- $\mathsf{Sub}(B)$= $\mathsf{Sub}(A_0) \cup \ldots \cup \mathsf{Sub}(A_n) \cup \{B\}$,
- $\mathsf{TopRule}(B)$=$\mathsf{TopRule}(A)$,
- $\mathsf{DefRules}(B)$=$\mathsf{DefRules}(A)$,
- $\mathsf{LDefRules}(B)$=$\mathsf{LDefRules}(A)$

We call Pref(B) the set of preferences $\{\mathsf{Conc}(A_0), \ldots, \mathsf{Conc}(A_m)\}$. For every argument that it is not a Preference Argument we have Pref(A)=$\emptyset$. We also call $GenP(A)$ the set of preference arguments built from argument A.

We can compare arguments ($>$) using one of the standard ASPIC$^+$ orderings:

Definition 21 (Aspic$^+$ Orderings). *Given two arguments A and B, we say that B is strictly preferred to A ($B \succ A$) if either:*

- ***Last Link**: $\mathsf{DefPrem}(A) \lhd \mathsf{DefPrem}(B)$ if both $\mathsf{LDefRules}(A)$ and $\mathsf{LDefRules}(B)$ are empty, else $\mathsf{LDefRules}(A) \lhd \mathsf{LDefRules}(B)$;*

- ***Weakest Link**: $\mathsf{DefPrem}(A) \lhd \mathsf{DefPrem}(B)$ ($\mathsf{DefRules}(A) \lhd \mathsf{DefRules}(B)$) if both $\mathsf{DefRules}(A)$ and $\mathsf{DefRules}(B)$ ($\mathsf{DefPrem}(A)$ and $\mathsf{DefPrem}(B)$) are empty, else $\mathsf{DefPrem}(A) \lhd \mathsf{DefPrem}(B)$ and $\mathsf{DefRules}(A) \lhd \mathsf{DefRules}(B)$*

where $X \lhd Y$ is either:

- ***Elitist**: $\exists x \in X$ s.t. $\forall y \in Y, \mathsf{sup}(y, x)$*

- ***Democrat**: $\forall x \in X$ s.t. $\exists y \in Y, \mathsf{sup}(y, x)$*

We can then build defeat over the attack definition (as specified in Definition 8):

Definition 22 (Direct Defeat). *An argument A directly defeats an argument B if:*

- *A directly rebuts/undermines B, and*
  - *$B \not> A$ w.r.t. the preferences in $Pref(B)$ or*
  - *it is not the case that $\mathsf{Conc}(B) \rhd \mathsf{Conc}(A)$*
- *A directly undercuts B*

We can now define our argumentation framework that will be assessable with Dung's standard semantics.

Definition 23 (Preference Argumentation Framework). *Let AT be an argumentation theory $< AS, K >$. An abstract argumentation framework defined by AT, is a tuple $< \mathcal{A}, \rightsquigarrow >$ where:*

- *$\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$ with $\mathcal{A}'$ the set of all arguments constructed from AT according to Definition 13 and $\mathcal{A}''$ the set of all arguments constructed from AT and $\mathcal{A}'$ according to Definition 20;*

- *for any arguments X and $Y \in \mathcal{A}$, $X \rightsquigarrow Y$ iff X directly defeats $Y' \in Sub(Y)$ (Definition 8 and Definition 22) and $\not\exists Y'_p \in GenP(Y')$ s.t. $Pref(Y'_p) \subseteq Pref(Y) \wedge Y'_p > X$*

Example 3 (Preference Example)**.** *In this example we use standard negation as source for conflicts, i.e., for every* $\phi \in L_p$, $\phi \rhd \neg\phi$ *and* $\neg\phi \rhd \phi$*. Let us take the theory where* $K_p = \{a, b, c, \neg r, \mathsf{sup}(p,q), \mathsf{sup}(q,h)\}$, $K_s = \{\mathsf{usable}(p), \mathsf{usable}(q), \mathsf{usable}(r)\}$ *where* $p : a \Rightarrow d$, $q : b \Rightarrow \neg d$ *and* $r : c \Rightarrow \mathsf{sup}(p,g)$*, and arguments (Definition 13):*

$$\begin{array}{llll}
\texttt{A0} : \mathsf{usable}(p) & & \texttt{A6} : \mathsf{sup}(p,q) \\
\texttt{A1} : \mathsf{usable}(q) & & \texttt{A7} : \mathsf{sup}(q,h) \\
\texttt{A2} : \mathsf{usable}(r) & & \texttt{A8} : \texttt{A0, A3} \Rightarrow d \\
\texttt{A3} : a & & \texttt{A9} : \texttt{A1, A4} \Rightarrow \neg d \\
\texttt{A4} : b & & \texttt{A10} : \texttt{A2, A5} \Rightarrow \mathsf{sup}(p,g) \\
\texttt{A5} : c & & \texttt{A11} : \neg r
\end{array}$$

Starting from these arguments, we can build the following set of Preference Arguments (Definition 20):

$$\begin{array}{l}
\texttt{P0A8} : \texttt{A10, A0, A3} \Rightarrow d \\
\texttt{P1A8} : \texttt{A6, A0, A3} \Rightarrow d \\
\texttt{P2A8} : \texttt{A6, A10, A0, A3} \Rightarrow d \\
\texttt{P0A9} : \texttt{A7, A1, A4} \Rightarrow \neg d
\end{array}$$

P0A8, P1A8 and P2A8 represent the link between A8 and its related preferences: $\mathsf{sup}(p,g)$ *and* $\mathsf{sup}(p,q)$*. Similarly, P0A9 connects* $\mathsf{sup}(q,h)$ *to A9. The last step before the evaluation of the framework is to determine attacks (Definition 8) and defeats (Definition 22). Last-Link Eli (Definition 21) is our selected ordering.*

- *A11 attacks and defeats A10, P0A8 and P2A8;*
- *A8 attacks and defeats A9;*
- *A8 attacks and defeats PA09—*$\mathsf{sup}(q,h)$ *is not relevant;*
- *A9 attacks and defeats A8;*
- *A9 attacks and defeats P0A8—*$\mathsf{sup}(p,g)$ *is not relevant;*
- *A9 attacks and does not defeat P1A8—A8* $\succ$ *A9 according to* $\mathsf{sup}(p,q)$*;*
- *A9 attacks and does not defeat P2A8—A8* $\succ$ *A9 according to* $\mathsf{sup}(p,q)$*;*

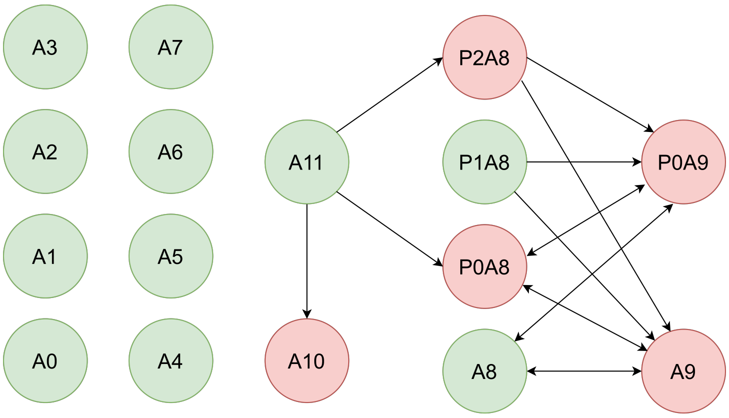


Figure 3.2: Argumentation framework from Example 3

- *same behaviour in other cases*

Figure 3.2 shows the resulting framework evaluated under grounded semantics. $A10$ successfully undercuts $A10$, thus disqualifying the preference $\mathsf{sup}(p,g)$ and the arguments relying on it ($P0A8$ and $P2A8$). However, $A8$ can still use the preference $\mathsf{sup}(p,q)$ to refute $A9$, and then $P0A9$. The statement $\mathsf{sup}(q,h)$ is not relevant to the case in exam end does not help $A9$'s case.

Let us now examine the connection between the Preference-based argumentation framework and standard ASPIC by showing that what is true according to the preference-based framework remains true when the verified preferences are applied a priori as in the original ASPIC$^+$ model.

Proposition 3. *Given the grounded extension E of the finitary Preference Argumentation Framework PAF=$<\mathcal{A},\rightsquigarrow>$ built from the argumentation theory AT=$<<L_p, R, n, >, K>$ with R_s closed under contraposition and Θ the selected ordering, we define the set of valid preferences P_v as $\{\mathsf{sup}(r_0, r_1) | \exists a \in E s.t. \mathsf{Conc}(a) = \mathsf{sup}(r_0, r_1)\}$. Using AT we can derive the standard argumentation framework $<\mathcal{A}', \rightsquigarrow'>$ according to the preferences P_v and the ordering Θ using standard ASPIC$^+$ definitions. The resulting grounded extension E' is such that $E' = E \cap \mathcal{A}'$.*

The full proof follows.

Proposition 3. Let's consider an argumentation framework $PAF =< \mathcal{A}, \rightsquigarrow >$. We call the characteristic function of PAF the function $F : 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ such that $F(Args) = \{X | \forall Y$ such that $Y \rightsquigarrow X$, then $\exists Z \in Args$ such that $Z \rightsquigarrow Y\}$ where $Args \subseteq \mathcal{A}$. Let us consider grounded extension as the smallest fixed point of the characteristic function F—i.e. the union of a sequence $E_0, \ldots, E_n$ obtained by iterative application of the F function on the empty set, and where $E_0 = \emptyset$. We prove that $E \cap \mathcal{A}' = E'$.

We first prove that $E \cap \mathcal{A}' \subseteq E'$. Suppose $a \in E \cap \mathcal{A}'$. We prove that $a \in E'$ as follows.

Base case: a has no defeaters in $\mathcal{A}$ according to $\rightsquigarrow$ so $a \in E_1 \cap \mathcal{A}'$. By construction, there can only be defeaters of a in $\mathcal{A}'$ according to $\rightsquigarrow'$ if there is at least a defeater of a in $\mathcal{A}$ according to $\rightsquigarrow$. But that is not the case, so a has no defeaters in $\mathcal{A}'$, so $a \in E'$.

Induction step: Assume that all arguments in $E_{i-1} \cap \mathcal{A}'$ are in E'. Take any $a \in E_i \cap \mathcal{A}'$. Let's consider any argument b such that b defeats a according to $\rightsquigarrow$. We have three cases: b does not exists in $\mathcal{A}'$ since it is a preference argument, then it can not defeat a according to $\rightsquigarrow'$; b exists in $\mathcal{A}'$ but it does not defeat a according to $\rightsquigarrow'$ – that is the case in which exist a set of preferences (represented by their arguments in $\mathcal{A}$) that are in E and make the attack from b to c invalid; b exists in $\mathcal{A}'$ and it still defeats a according to $\rightsquigarrow'$. Dy definition, we have that for any argument b such that $b \rightsquigarrow a$, it must exist an argument $c \in E_{i-1}$ such that $c \rightsquigarrow b$. By induction hypothesis we know that any such $c \in E_{i-1} \cap \mathcal{A}'$ is also in E'. The questions are whether c is in $\mathcal{A}'$ and if it still defeats b according to $\rightsquigarrow'$. As for the first question, the cases are two: c is a standard argument (built using Definition 6), then by construction it is also in $\mathcal{A}'$; c is a preference argument, hence not in $\mathcal{A}'$, but in that case there would exist a standard argument $c' \in E_{i-1} \cap \mathcal{A}'$ having the same defeat relations as c—it would be the original argument on which the preference argument was built (Definition 20). As for the second question, c does not defeat b in $\rightsquigarrow'$ only if it exists a combination of preferences that makes the attack from c to b invalid. But that would mean having a preference argument b'_p in $\mathcal{A}$, based on $b' \in \mathsf{Sub}(p)$, representing the same combination of preferences. But in that case would not be possible for c to defeat b'_p, conversely it would be defeated by it (or, in the case of c having a strict top rule, by an argument d built by the contraposition of c's top rule, thanks to the closure under contraposition and the reasonability of the ordering). But this contradicts the fact that c belongs to E_{i-1}, hence c does defeat b in $\rightsquigarrow'$. The three cases together prove that $a \in E'$.

We next prove that $E' \subseteq E \cap \mathcal{A}'$. Suppose $a \in E'$. We prove that $a \in E \cap \mathcal{A}'$ as follows.

Base case: a has no defeaters in $\mathcal{A}'$ so $a \in E'_1$. Then either a has not attackers in $\mathcal{A}'$, or it has one or more attackers in $\mathcal{A}'$. As for the first case, let us suppose that exists an argument $b \in \mathcal{A}$ s.t. $(b, a) \in \rightsquigarrow$. Then, by construction, the attack would also exists in $\rightsquigarrow'$, but this contradicts the fact that a has no attackers in $\mathcal{A}'$. So a has no attackers in $\mathcal{A}$. We must check the latter case. Let's consider an argument $b \in \mathcal{A}'$ such that $(b, a) \in \rightsquigarrow$ but $\notin \rightsquigarrow'$. That means that exists a set of preferences $P_a = \{\mathsf{sup}(p_0, p_1), \ldots, \mathsf{sup}(p_{n-1}, p_n)\}$, that makes $a \succ b$. By construction, there must be in E a group of arguments $p_0, \ldots, p_n$ such that $\mathsf{Conc}(p_0) = \mathsf{sup}(p_0, p_1), \ldots, \mathsf{Conc}(p_n) = \mathsf{sup}(p_{n-1}, p_n)$. Also, we should have an argument $a_p \in \mathcal{A}$ having the same structure as a but also including $p_0, \ldots, p_n$ and their subarguments. Since $a \succ b$ according to P_a, then also $a_p \succ b$. We must determine if $(a_p, b) \in \rightsquigarrow$. There are two cases, either $\mathsf{TopRule}(b)$ is defeasible or it is strict. In the first case, $(a_p, b) \in \rightsquigarrow$ and $(b, a_p) \notin \rightsquigarrow$. Accordingly, to determine if $a \in E$, we must find out if $a_p \in E$. Since $p_0, \ldots, p_n \in E$ and b can not attack a_p, the only arguments we are interested in to determine its state are a's attackers (b excluded) since a_p has the same structure as a. In the latter case, by contraposition and reasonability of the ordering [Modgil and Prakken, 2013], we know that must exists an argument b_p^C obtained by contraposition of $\mathsf{TopRule}(b)$ s.t. $b_p^C > b$ and $(b_p^C, b) \in \rightsquigarrow$. By construction, we know that b_p^C shares all its direct subarguments with b, with the exception of a and $p_0, \ldots, p_n$. Since an attack from an argument in E to one of the common subs would also lead to b's exclusion from E, and by Definition 5.5 $(b, b_p^C) \notin \rightsquigarrow$, also in this case, we must find out if a_p is in E. Assume the existence of an argument c such that (c, a) and $(c, a_p) \in \rightsquigarrow$. There are two cases: c defeats a also according to $\rightsquigarrow'$, but that contradicts the fact that a does not have defeaters in $\rightsquigarrow'$; or c does not defeat a in $\rightsquigarrow'$. We are in the same situation we were for b, and then we can apply the same steps to find that exists an argument a'_p such that $a'_p \succ b$ and $a'_p \succ c$. Again, we would need a new a's attacker, let's call it d, to prove that a is not in E. We could iterate the same process again and again, but since the PAF is finitary, eventually we will not find a new a's attackers to test, hence we would have an argument a_p^n that defeats all a's defeaters, and it is defeated by none. So, $a \in E$.

Induction step: Assume that all arguments in E'_{i-1} are in E. Consider any $a \in E'_i$. Let's consider any argument b such that b defeats a according to $\rightsquigarrow'$. By definition, such b is defeated by E'_{i-1}. Since $\rightsquigarrow' \subseteq \rightsquigarrow$, and, by induction hypothesis, all arguments in E'_{i-1} are in E, we have that at least one of b's defeaters (according to $\rightsquigarrow$) belongs to E. There is another case to consider: it exists an argument $b \in \mathcal{A}'$ such that $(b, a) \in \rightsquigarrow$ but $\notin \rightsquigarrow'$. In this scenario, b could prevent a to be included in the extension. The proof goes exactly as in the base case: we recursively introduce a new preference argument a_p^n for every a's defeater. Eventually, we will be in the situation where there are no new defeaters to introduce (thanks to the finiteness of the framework). Then we would have an argument a_p^n that defeats all a's defeaters in $\mathcal{A}$, and it is defeated by none. The two cases together prove that $a \in E$.

Since $E \cap \mathcal{A}' \subseteq E'$ and $E' \subseteq E \cap \mathcal{A}'$ then $E' = E \cap \mathcal{A}'$. □

3.4 Mixing Everything Together

The introduced mechanism for handling conflicts within our framework presents an opportunity for further refinement, particularly by considering the insights gained from the developments made in the area of preferences. By revisiting and possibly revising the conflict mechanism in light of the advancements in preference handling, we aim to achieve a more cohesive and intuitive model.

To recap, in ASPIC$^+$, the definition of an argumentation system also includes the specification of the contrary relation $\rhd$. In our framework, on the contrary, conflicts have to be supported by arguments. More exactly, for an argument A to be able to directly attack another argument B, A must claim that its conclusion collides with (is a contrary to) the conclusion of B.

We can new define the notion of a conflict argument, namely, an argument that includes a subargument according to which its conclusion ϕ is contrary to a formula ψ:

Definition 24 (Conflict Argument)**.** *Let AT be an argumentation theory* $<$

$AS, K >$ based on a conflict language, and let $\mathcal{A}$ a set of arguments constructed from AT. Given $A, A_0 \in \mathcal{A}$ s.t. $\mathsf{Conc}(A_0) = \mathsf{contr}(\phi, \psi)$, a conflict argument B has one of the following forms:

- $A_0, A_1, \dots, A_n \Rightarrow \phi$ *if* $A = A_1, \dots, A_n \Rightarrow \phi$,
- $A_0, A_1, \dots, A_n \rightarrow \phi$ *if* $A = A_1, \dots, A_n \rightarrow \phi$,
- $A_0 \Rightarrow \phi$ *if* $A = \phi$ *and* $\phi \in K_p$,
- $A_0 \rightarrow \phi$ *if* $A = \phi$ *and* $\phi \in K_s$,

where:

- *Prem(B)=Prem(A),*
- *Conc(B)=Conc(A),*
- *Sub(B)=Sub$(A_0) \cup \ldots \cup$ Sub$(A_n) \cup \{B\}$,*
- *TopRule(B)=TopRule(A),*
- *DefRules(B)=DefRules(A),*
- *LDefRules(B)=LDefRules(A)*

Conflict arguments are used to generate attacks. In fact, to attack argument B, an argument A must claim (through a subargument) that its conclusion collides with B—i.e. A has to be a conflict argument:

Definition 25 (Conflict Attack)**.** *Let $A_1 \in \mathsf{Sub}(A)$ and $\mathsf{Conc}(A_i) = \mathsf{contr}(\phi, \psi)$, Then:*

- *A directly rebuts argument B iff $\mathsf{Conc}(B) = \psi$ and $\mathsf{TopRule}(B) \in R_d$;*
- *A directly undermines argument B iff $B \in K_p$ and $\mathsf{Conc}(B) = \psi$;*
- *A directly undercuts argument B iff $n(\mathsf{TopRule}(B)) = \psi$ and $\mathsf{TopRule}(B) \in R_d$;*

A directly attacks B if A directly rebuts, directly undermines or directly undercuts B.

Based on this notion of an attack (conflict attack) we can build a corresponding argumentation framework (conflict argumentation framework):

Definition 26 (Conflict Argumentation Framework). *Let AT be an argumentation theory $< AS, K >$ based on a conflict language. An abstract argumentation framework defined by AT, is a tuple $< \mathcal{A}, \rightsquigarrow >$ where:*

- *$\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$ with $\mathcal{A}'$ the set of all arguments constructed from AT according to Definition 13 and $\mathcal{A}''$ the set of all arguments constructed from AT and $\mathcal{A}'$ according to Definition 24;*

- *for any arguments X and $Y \in \mathcal{A}$, $X \rightsquigarrow Y$ iff X directly attacks $Y' \in Sub(X)$ (Definition 25)*

Example 4 (Conflict Example). *Let us consider again Example 1. In that case we make use of the standard definition of negation and undercutting, i.e., an argument for $\neg p$ directly undercuts the argument having top rule p. Conversely, in the Conflict-based Framework, the undercutting argument must support the claim that its conclusion collides with the rule identified by p. This may be the case not only when the undercutting argument has conclusion $\neg p$, but also when it has a different conclusion, let's take q for example, which is claimed to be in conflict with p.*

Let's try to reproduce the same result using a Conflict-based Argumentation Framework. For starters, we replace the $\neg p$ formula with a statement with no negation operators, namely q.

Accordingly, we build the theory where $K_p = \{a, c, d\}$, $K_s = \{\mathsf{usable}(p), \mathsf{usable}(q), \mathsf{usable}(r)\}$ where $p : a \Rightarrow b$, $q : c \Rightarrow q$ and $r : d \Rightarrow \mathsf{contr}(q, p)$, and arguments (Definition 6):

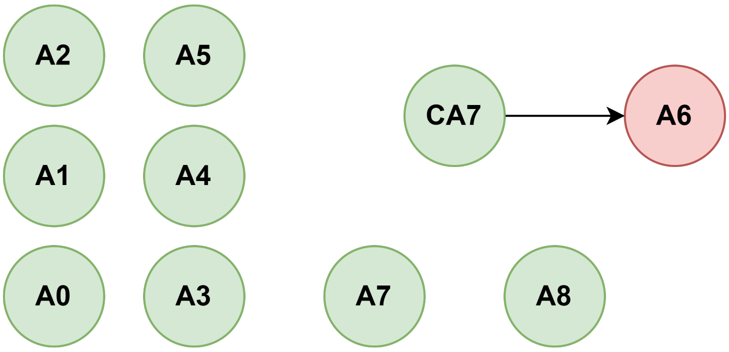


Figure 3.3: Argumentation framework from Example 4

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\mathtt{A0} & : & \textsf{usable}(p) & \mathtt{A5} & : & d \\
\mathtt{A1} & : & \textsf{usable}(q) & \mathtt{A6} & : & \mathtt{A0},\ \mathtt{A3} \Rightarrow b \\
\mathtt{A2} & : & \textsf{usable}(r) & \mathtt{A7} & : & \mathtt{A1},\ \mathtt{A4} \Rightarrow q \\
\mathtt{A3} & : & a & \mathtt{A8} & : & \mathtt{A2},\ \mathtt{A5} \Rightarrow \textsf{contr}(q,p) \\
\mathtt{A4} & : & c & & &
\end{array}$$

Starting from these arguments, we can build a single Conflict Argument (Definition 24), namely $CA7 : A8, A1, A4 \Rightarrow q$*. This argument attacks any argument using the* p *formula (either as a rule, conclusion or premise), since it supports conclusion* q *and it includes the subargument* $A8$ *according to which* q *collides with* p*. In our case* $CA7$*'s attack is directed against* $A6$*, the only argument using the rule identified by* p*. As in Example 1,* $A6$ *(*b*) is rejected according to grounded (and all derived) semantics. It should be noted how the attack is now dependant on the info available in the argumentation theory: without the premise* a *it would have been impossible to derive the conflict relating* q *and* p*, and consequently to build* $CA7$ *and its undercutting attack. Moreover, the attack would have been ineffective also in the case of exclusion of* $A3$ *or* $A8$ *from the extension.*

The presented Conflict-based Argumentation Framework can be analysed w.r.t. the one introduce in Section 3.2. Indeed, the two models share the same motivating idea and most of their inner mechanism. In particular, their main difference is in the definition of Conflict Arguments and Direct Attack Arguments. Their role in the two frameworks is the same and their definition is also quite similar, if not for two details: *i)* their form, i.e., Conflict Arguments share the structure with their base arguments, while Direct Attack Arguments include their base argument as a proper subargument; *ii)* their instantiation, i.e., Conflict Arguments are based on available conflicts (even if not applicable), while Direct Attack Arguments are only built if a target is available (they are target specific). In any case, these differences have an impact only on the size of the framework, but not in their behaviour, hence we refer the reader to the proofs provided in Section 3.2 w.r.t. the soundness of the extension under Dung's grounded semantics.

In this section we revisited the conflict mechanism to make it closer to the one introduced for preferences, but still a final effort is required in order to join the work on defeasible conflicts (Section 3.4) and the one on preferences (Section 3.3) in the same formalism. Indeed, Definition 22 requires a complete knowledge about the conflict relation in order to correctly identify contrary attacks and make them independent from preferences. In the presence of dynamic conflicts, having such kind of knowledge would be impossible since the final conformation of the conflict relation is known only after the evaluation of the framework. To better visualise the problem, assume a framework with two arguments claiming a and b, with b and a being in conflict between them and $b \succ a$. In the case both conflicts are confirmed ($b \rhd a$ and $a \rhd b$) the preference should be considered while building the defeat relation, and, consequently, the argument for b should defeat the one for a, but not vice-versa. The situation changes if, adding new knowledge to the system, we discover that $b \rhd a$ does not hold anymore—i.e., the argument concluding $\mathsf{contr}(b, a)$ is outside the target extension in the context of a Conflict Argumentation Framework. In such scenario the preference should be overruled, allowing a to attack and defeat b.

To cope with this new case we provide a new definition of defeat applicable in presence of dynamic conflicts:

Definition 27 (Direct Defeat with dynamic preferences)**.** *An argument A directly defeats an argument B if:*

- *A directly rebuts/undermines B, and*
  - *$B \not\succ A$ w.r.t. the preferences in $Pref(B)$ or*
  - *it is not the case that $\exists B' \in Sub(B)$ s.t. $\mathit{Conc}(B') = \mathit{contr}(\mathit{Conc}(B), \mathit{Conc}(A))$*

- *A directly undercuts B*

Intuitively, preferences are taken into account only if the attacked argument is a potential defeater of the attacking argument—i.e., it exists a conflict argument relating attacked and attacking arguments between the subarguments of the former.

At last, we can define our *Meta-Argumentation* framework as:

Definition 28 (Meta-Argumentation Framework)**.** *Let AS_m be an argumentation system $< L_{cf}, R, n, \emptyset >$, where L_{cf} is a language based on Definition 12, Definition 14 and Definition 19, and AT_m be an argumentation theory $< AS_m, K >$. An abstract argumentation framework defined by AT_m, is a tuple $< \mathcal{A}, \rightsquigarrow >$ where:*

- *$\mathcal{A} = \mathcal{A}' \cup \mathcal{A}'' \cup \mathcal{A}'''$ with*
  - *$\mathcal{A}'$ is the set of all arguments constructed from AT_m using Definition 13;*
  - *$\mathcal{A}''$ is the set of all arguments constructed from AT_m and $\mathcal{A}'$ using Definition 24;*
  - *$\mathcal{A}'''$ is the set of all arguments constructed from AT_m and $\mathcal{A}''$ using Definition 20*
- *for any arguments X and $Y \in \mathcal{A}$, $X \rightsquigarrow Y$ iff X directly defeats $Y' \in \mathsf{Sub}(Y)$ according to Definition 25 and Definition 27 and $\not\exists Y'_p \in GenP(Y')$ s.t. $Pref(Y'_p) \subseteq Pref(Y) \wedge Y'_p > X$*

We can evaluate any Meta-argumentation framework using Dung's grounded semantics.

3.5 Reasoning with Burden of Persuasion

In this section of the thesis, we turn our attention to the Burden of Persuasion, a concept deeply embedded in both our formal argumentation model and the realm of legal reasoning. This concept, fundamental to the legal process, dictates which party in a dispute must prove their point to a certain standard, shaping the dynamics of legal argumentation.

We introduce a meta-argumentation framework specifically tailored to address the Burden of Persuasion. This specialised framework not only embeds the Burden of Persuasion at its core but also aligns seamlessly with established legal methodologies, offering a refined and intricate approach to the evaluation of arguments within legal contexts. By integrating this concept, our framework bridges the gap between theoretical argumentation models and practical legal reasoning, ensuring a more realistic and applicable tool for legal practitioners.

Throughout this section, we will enrich our discussion with practical examples from the legal field. These examples are carefully chosen to illustrate the efficacy of our meta-argumentation framework in real-world legal scenarios, demonstrating how it adeptly handles the complexities and nuances associated with the Burden of Persuasion in legal disputes. This approach not only provides clarity on the theoretical aspects of our model but also showcases its direct applicability and relevance in the legal domain.

For the sake of simplicity we choose to model our meta-argumentation framework by exploiting bimodal graphs, which are often exploited to both define meta-level concepts and understand the interactions of object-level and meta-level arguments [Ogunniye et al., 2018, Müller et al., 2013]. Accordingly, Subsection 3.5.1 presents the object-level argumentation language exploited by our model, leveraging on an ASPIC$^+$-like argumentation framework [Prakken, 2010]. Then, the meta-level argumentation language based on the use of argument schemes [Walton et al., 2008] is introduced in Subsection 3.5.2.

3.5.1 Object-level argumentation

In our analysis, while we utilize the ASPIC+$^+$ framework, it's important to note that we are adopting a slightly simplified version. This approach concentrates exclusively on defeasible rules and premises, with strong negation being identified as the main source of conflicts. The subsequent definitions will further elucidate this adjusted framework.

Let a literal be an atomic proposition or its negation.

Notation 2. *For any literal ϕ, its complement is denoted by $\bar{\phi}$. That is, if ϕ is a proposition p, then $\bar{\phi} = \neg p$, whereas if ϕ is $\neg p$, then $\bar{\phi}$ is p.*

Let us also identify burdens of persuasion, i.e., those literals whose proof requires a convincing argument. We assume that such literals are consistent (it cannot be the case that there is a burden of persuasion on both ϕ and $\bar{\phi}$).

Definition 29 (Burdens of persuasion)**.** *Burdens of persuasion are represented by predicates of the form bp(ϕ), stating the burden is allocated on the literal ϕ.*

Literals are put in relation with *bp* predicates through defeasible rules.

Definition 30 (Defeasible rule)**.** *A **defeasible rule** r has the form:*

$$\rho: \quad \phi_1, ..., \phi_n, \sim\phi'_1, ..., \sim\phi'_m \Rightarrow \psi$$

with $0 \leq n, m$*, and where*

- *ρ is the unique identifier for r, denoted by $\mathsf{N}(r)$;*
- *each $\phi_1, \ldots, \phi_n, \phi'_1, \ldots, \phi'_m, \psi$ is a literal or a* bp *predicate;*
- *$\phi_1, \ldots \phi_n, \sim\phi'_1, ..., \sim\phi'_m$ are denoted by Antecedent(r);*
- *ψ is denoted by Consequent(r);*
- *$\sim\phi$ denotes the weak negation (negation by failure) of ϕ—i.e., ϕ is an exception that would block the application of the rule whose antecedent includes $\sim\phi$.*

The unique identifier of a rule can be used as a literal to specify that the named rule is applicable, and its negation to specify that the rule is inapplicable, dually [Modgil and Prakken, 2014b].

A superiority relation $\succ$ is defined over rules: $s \succ r$ states that rule s prevails over rule r.

Definition 31 (Superiority relation)**.** *A **superiority relation** $\succ$ over a set of rules Rules is a transitive, antireflexive and antisymmetric binary relation over Rules.*

A defeasible theory consists of a set of rules and a superiority relation over the rules.

Definition 32 (Defeasible theory)**.** *A **defeasible theory** is a tuple* $\langle Rules, \succ\rangle$ *where Rules is a set of rules, and* $\succ$ *is a superiority relation over Rules.*

Given a defeasible theory, we can construct arguments by chaining rules from the theory [Modgil and Prakken, 2014b, Caminada and Amgoud, 2007b, Vreeswijk, 1997b].

Definition 33 (Argument)**.** *An **argument** A constructed from a defeasible theory* $\langle Rules, \succ\rangle$ *is a finite construct of the form:* $A : A_1, \dots A_n \Rightarrow_r \phi$ *with* $0 \leq n$*, where*

- *A is the argument's unique identifier;*
- *$A_1, \dots, A_n$ are arguments constructed from the defeasible theory* $\langle Rules, \succ\rangle$*;*
- *ϕ is the* conclusion *of the argument, denoted by* $\mathsf{Conc}(A)$*;*
- $r : \mathsf{Conc}(A_1), \dots, \mathsf{Conc}(A_n) \Rightarrow \phi$ *is the top rule of A, denoted by* $\mathsf{TopRule}(A)$*.*

Notation 3. *Given an argument* $A : A_1, \dots A_n \Rightarrow_r \phi$ *as in Definition 33,* $\mathsf{Sub}(A)$ *denotes the set of subarguments of A, i.e.,* $\mathsf{Sub}(A) = \mathsf{Sub}(A_1) \cup \dots \cup \mathsf{Sub}(A_n) \cup \{A\}$*.* $\mathsf{DirectSub}(A)$ *denotes the direct subarguments of A, i.e.,* $\mathsf{DirectSub}(A) = \{A_1, \dots, A_n\}$*.*

Preferences over arguments are defined via a last-link ordering: argument A is preferred over argument B if the top rule of A is stronger than the top rule of B.

Definition 34 (Preference relation)**.** *A **preference relation** $\succ$ is a binary relation over a set of arguments $\mathcal{A}$: argument A is preferred to argument B – denoted by $A \succ B$ – iff* $\mathsf{TopRule}(A) \succ \mathsf{TopRule}(B)$*.*

Arguments are put in relation with each others according to the attack relation.

Definition 35 (Attack)**.** *Argument A **attacks** argument B iff A undercuts or rebuts B, where*

- *A undercuts B (on B') iff* $\mathsf{Conc}(A) = \neg \mathsf{N}(\rho)$ *for some* $B' \in \mathsf{Sub}(B)$*, where ρ is* $\mathsf{TopRule}(B')$

- *A rebuts B (on B') iff either* (i) $\mathsf{Conc}(A) = \bar{\phi}$ *for some* $B' \in \mathsf{Sub}(B)$ *of the form* $B''_1, ..., B''_M \Rightarrow \phi$ *and* $B' \not\prec A$, *or* (ii) $\mathsf{Conc}(A) = \phi$ *for some* $B' \in \mathsf{Sub}(B)$ *such that* $\sim\phi \in Antecedent(\mathsf{TopRule}(B'))$

In short, arguments can be attacked either on a conclusion of a defeasible inference (*rebutting* attack) or on a defeasible inference step itself (*undercutting* attack).

Definition 36 (Argumentation graph)**.** *An **argumentation graph** is a tuple* $\langle \mathcal{A}, \rightsquigarrow \rangle$*, where* $\mathcal{A}$ *is the set of all arguments, and* $\rightsquigarrow$ *is* attack *relation over* $\mathcal{A}$*.*

Notation 4. *Given an argumentation graph* $G = \langle \mathcal{A}, \rightsquigarrow \rangle$*, we write* $\mathcal{A}_G$ *and* $\rightsquigarrow_G$ *to denote the graph's arguments and attacks, respectively.*

Now, let us introduce the notion of the $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$-labelling of an argumentation graph, where each argument in the graph is labelled IN, OUT, or UND, depending on whether it is accepted, rejected, or undecided, respectively.

Definition 37 (Labelling)**.** *Let G be an argumentation graph. An* $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$*-**labelling** L of G is a total function* $\mathcal{A}_G \rightarrow \{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$*.* $\mathcal{L}(\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}, G)$ *denotes the set of all* $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$*-**labellings** of G.*

A labelling-based semantics prescribes a set of labellings for any argumentation graph according to some criterion embedded in its definition.

Definition 38 (Labelling-based semantic)**.** *Let G be an argumentation graph. A labelling-based semantics S associates with G a subset of* $\mathcal{L}(\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}, G)$*, denoted as* $L_S(G)$*.*

3.5.2 Meta-level argumentation

A fundamental aspect to consider when dealing with a multi-level argumentation graph is how the higher-level graphs can be built starting from the object-level ones. For the purpose, in this work – following the example in [Ogunniye et al., 2018] – we leverage on argument schemes [Walton et al., 2008]. In short, argumentation schemes are commonly-used patterns of reasoning. They can be formalised in a rule-like form [Prakken, 2005] where every argument scheme consists of a set of conditions and a conclusion. If the conditions are met, then the conclusion holds. Each scheme comes with a set of *critical questions* (CQ), identifying possible exceptions to the admissibility of arguments derived from the schemes.

Definition 39 (Meta-predicate)**.** *A meta-predicate P_M is a symbol that represents a property or a relation between object-level arguments. Let be $\mathcal{M}$ the set of all P_M.*

Definition 40 (Object-relation meta-predicate)**.** *An object-relation meta-predicate O_M is a predicate stating the existence of a relation at the object level—e.g., attacks, preferences, conclusions. Let be $\mathcal{O}$ the set of all O_M.*

Moving from the above definitions we can define an argument scheme as:

Definition 41 (Argument Scheme)**.** *An* ***argument scheme*** *s has the form:*

$$s : P_1, ..., P_n, \sim P'_1, ..., \sim P'_m \Rightarrow Q$$

with $0 \leq n, m$, and where

- *each $P_1, \ldots, P_n, P'_1, \ldots, P'_m \in \mathcal{M} \cup \mathcal{O}$, while $Q \in \mathcal{M}$*
- *$\sim P$ denotes weak negation (negation by failure) of P—i.e., P is an exception that would block the application of the rule whose antecedent includes $\sim P$*
- *we denote with CQ_s the set of critical questions associated to scheme s.*

Using argument schemes we can build meta-arguments.

Definition 42 (Meta-Argument)**.** *A* ***meta-argument*** *A constructed from a set of argument schemes S and an object-level argumentation graph G is a finite construct of the form: $A : \; A_1, \ldots A_n \Rightarrow_s P$ with $0 \leq n$, where*

- *A is the argument's unique identifier;*
- *$s \in S$ is the scheme used to build the argument;*
- *$A_1, \ldots, A_n$ are arguments constructed from S and G;*
- *P is the* conclusion *of the argument, denoted by $\mathsf{Conc}(A)$.*

$CQ(A)$ denotes the critical questions associated to scheme s. The same notation introduced for standard arguments in Notation 3 also applies to meta-arguments.

We can now define attacks over meta-arguments, or, *meta-attacks*.

Definition 43 (Meta-Attack)**.** *An argument A* ***attacks*** *argument B (on B') iff either* (i) $\mathsf{Conc}(A) = \bar{P}$ *for some* $B' \in \mathsf{Sub}(B)$ *of the form* $B_1'', ..., B_M'' \Rightarrow P$*, or* (ii) $\mathsf{Conc}(A) = P$ *for some* $B' \in \mathsf{Sub}(B)$ *such that* $\sim P \in Antecedent(\mathsf{TopRule}(B'))$*.*

The same definition of *argumentation graph* and *labellings* introduced for standard argumentation in Definitions 36, 37, 38 also holds for meta-arguments and for the meta level.

3.5.3 Burden of persuasion as meta-argumentation

Informally, we can say that when we talk about the notion of the burden of persuasion concerning an argument, we intuitively argue over that argument according to a meta-argumentative approach.

Let us consider, for instance, an argument A: if we allocate the burden over it, we implicitly impose the duty to prove its admissibility on A. Thus, moving the analysis up to the meta level of the argumentation process, it is like having two arguments, let them be F_{BP} and S_{BP}, reflecting the burden of persuasion status. According to this perspective, F_{BP} states that "the burden is not satisfied if A *fails* to prove its admissibility" – i.e. A should be rejected or undefined – and, of course, F_{BP} is not compatible with A being accepted. Alongside, S_{BP} states that "A is acceptable since it *satisfies* its burden". F_{BP} and S_{BP} have a contrasting conclusion and thus they attack each other.

Analysing the burden from this perspective makes immediately clear that the notions that the meta model should deal with are:

N1 the notion of the burden itself expressing the possibility for an argument to be allocated with a burden of persuasion (i.e., *burdened argument*)

N2 the possibility that this burden is satisfied (that is, a *burden met*) or not satisfied

N3 the possibility of making *attacks* involving burdened arguments ineffective.

The outline of that multi-part evaluation scheme for burdens of persuasion in argumentation is now visible and can be formally designed. In the following, we formally define these concepts by exploiting bimodal argument graphs as techniques for expressing the two main levels of the model – meta and object level – and the relationships between the two.

In particular, we are going to define each set of the bimodal argument graph tuple $\langle \mathcal{A}_O, \mathcal{A}_M, \mathcal{R}_O, \mathcal{R}_M, \mathcal{S}_A, \mathcal{S}_R \rangle$. With respect to $\mathcal{A}_O$ and $\mathcal{R}_O$, representing respectively the set of object-level arguments and attacks, they are built accordingly to the argumentation framework discussed in Subsection 3.5.1. Hence, our analysis focuses on the meta-level graph $\langle \mathcal{A}_M, \mathcal{R}_M \rangle$ and on the support sets connecting the two levels ($\mathcal{S}_A$ and $\mathcal{S}_R$).

Meta-level graph

We now proceed to detail all the argumentation schemes used to build arguments in the meta-level graph. Every scheme comes along with its critical questions. As we will see in the next sections, all the critical questions have to be interpreted as kind of "presumptions": they are believed to be true during the construction and evaluation of the argumentation framework – i.e., they are not used as possible attack dimensions –, but their post hoc verification invalidates the entire solution.

Let us first introduce the basic argumentation scheme enabling the definition and representation of an argument with an allocation of the burden of persuasion (i.e., reifying **N1**). We say that an object-level argument A has the burden of persuasion on it if exists an object-level argument B such that $\mathsf{Conc}(B) = bp(\mathsf{Conc}(A))$. This notion is modelled through the following argument scheme:

$$\textit{conclusion}(A, \phi), \textit{conclusion}(B, bp(\phi)) \Rightarrow \textit{burdened}(A) \tag{S0}$$

$$\textit{Is argument B provable?} \tag{$\mathsf{CQ_{S0}}$}$$

where $bp(\phi)$ is a predicate stating ϕ is a literal with the allocation of the burden, $\textit{conclusion}(A, \phi)$ is a structural meta-predicate stating that $\mathsf{Conc}(A) = \phi$ holds, and $\textit{burdened}(A)$ is a meta-predicate representing the allocation of the burden on A. Clearly, an argument produced using this scheme only holds if both the arguments A and B on which the inference is based hold—critical question $\mathsf{CQ}_{\mathsf{S0}}$.

Analogously, we introduce the scheme S1 representing the absence of such an allocation:

$$conclusion(A, \phi) \Rightarrow \neg burdened(A) \tag{S1}$$

$$\textit{Is argument A provable? Are arguments concluding bp}(\phi) \textit{ not provable?} \tag{$\mathsf{CQ}_{\mathsf{S1}}$}$$

Then, as informally introduced at the beginning of this section, we have two schemes reflecting the possibility for a burdened argument to meet or not the burden (**N2**).

$$burdened(A) \;\Rightarrow\; bp\_met(A) \tag{S2}$$

$$burdened(A) \;\Rightarrow\; \neg bp\_met(A) \tag{S3}$$

$$\textit{Is argument A provable?} \tag{$\mathsf{CQ}_{\mathsf{S2}}$}$$

$$\textit{Is argument A always refuted or undecided?} \tag{$\mathsf{CQ}_{\mathsf{S3}}$}$$

where *bp_met* is the meta-predicate stating the burden has been met. It is important to notice that the two schemes above reach opposite conclusions from the same grounds—i.e., the presence of the burden on argument A. The discriminating elements are the critical questions they are accompanied by. In the case of S2, we have that only if a burden of persuasion on argument A exists, and A is acceptable ($\mathsf{CQ}_{\mathsf{S3}}$), then the burden is satisfied. On the other side, the validity of S3 is bound to the missing admissibility of argument A. We will see in Section 3.5.3 how the meta-arguments and the associated questions concur to determine the model results.

Let us now consider attacks between arguments and their relation with the burden of persuasion allocation. When a burdened argument fails to meet the burden, the only thing affecting the argument acceptability is the burden itself—i.e., attacks from other arguments do not influence the status of the burdened argument, which only depends on its inability to satisfy the burden. The same applies to attacks issued by an argument that fails to meet the burden: the failure implies the argument rejection and, as a direct consequence, the inability to effectively attack other arguments. In order to capture the nuance of discerning between effective and ineffective object-level attacks w.r.t. the concept of burden of persuasion (**N3**), we define the following scheme:

$$attack(B, A), \sim(\neg bp\_met(A)), \sim(\neg bp\_met(B)) \Rightarrow effectiveAttack(B, A) \qquad (\mathsf{S4})$$

Can we prove arguments A or B do not fail to meet their burden? $(\mathsf{CQ}_{\mathsf{S4}})$

where *attack* is a structural meta-predicate stating an attack relation at the object level, whereas *effectiveAttack* is a meta-predicate expressing that an attack should be taken into consideration according to the burden of persuasion allocation. In other words, if an object-level attack involves burdened arguments, and one of these fail to satisfy the burden, then the attack is considered not effective w.r.t. the allocation of the burden.

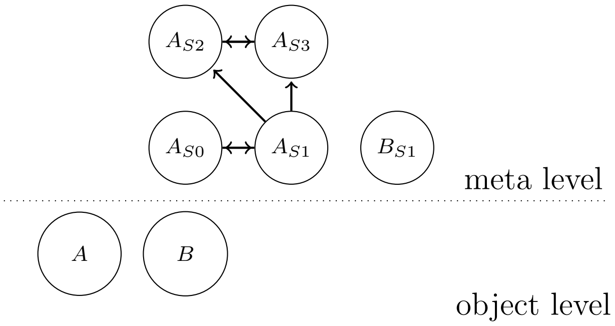
The aforementioned schemes can be used to create a meta-level graph containing all the information about constraints related to the burden of persuasion concept thus leading to a clear separation of concerns, as shown in the following example.

Example 5 (Base)**.** *Let us consider two object-level arguments A and B, concluding the literals a and $bp(a)$ respectively. Using the schemes in Subsection 3.5.3 we can build the following meta-level arguments:*

- A_{S0} *representing the allocation of the burden on argument A.*
- A_{S1} *and* B_{S1} *standing for the absence of a burden on arguments A and B respectively. The scheme used to build those arguments exploits weak negation in order to cover those scenarios where an argument concluding a* bp *literal exists at the object-level, but it is found not acceptable.*
- A_{S2} *and* A_{S3} *sustaining that* (i) *A was capable of meeting the burden on it,* (ii) *A was not capable of meeting its burden.*

The meta-level graph (Figure 3.4) points out the relations actually implicit in the notion of burden of persuasion over an argument, where, intuitively, we argue over the consequences of A's possibly succeeding/failing to meet the burden. At the meta level, all the possible scenarios can be explored by applying different semantics over the meta-level graph.

Considering for instance Dung's preferred semantics [Baroni et al., 2011c], we can obtain two distinct outcomes: (1) the burden is not satisfied, i.e., argument A_{S3} is accepted, and consequently, A_{S2} is rejected, or (2) we succeed in proving A_{S2}, i.e., the burden is met and A_{S3} is rejected (A_{S0}, A_{S1} are accepted and rejected accordingly). Although the example is really simple – only basic schemes for reasoning on the burden are considered at the meta-level – it clearly demonstrates the possibility of reasoning over the burdens, since, i.e., it establishes whether or not there is a burden on a literal ϕ – argument B in the example – and enables the evaluation of the consequences of a burdened argument to meet or not its burden.



Meta-level arguments:

$A_{S0} :\Rightarrow burdened(A)$
$A_{S1} :\Rightarrow \neg burdened(A)$
$A_{S2} : A_{S0} \Rightarrow bp\_met(A)$
$A_{S3} : A_{S0} \Rightarrow \neg bp\_met(A)$
$B_{S1} :\Rightarrow \neg burdened(B)$

Object-level arguments:

$A :\Rightarrow a$
$B :\Rightarrow bp(a)$

Figure 3.4: Object and meta level graphs from Example 5

Object- and meta-level connection: supporting sets

Let us now define how the meta level and the object level interact. Indeed, it is not enough to reason on the consequences of the burden of persuasion allocation only concerning the burdened argument, but the results of the argument satisfying or not such a burden constraint should affect the entire object-level graph. According to the standard bimodal graph theory, defining how the object level and the meta level interact is the role of the argument support relation $\mathcal{S}_A$ and of the attack support relation $\mathcal{S}_R$, respectively. According to Definition 10, every element at level n is connected to an argument at level $n+1$ by a support edge in $\mathcal{S}_A$ or $\mathcal{S}_R$, depending on whether it is either an argument or an attack.

Let us define the support set $\mathcal{S}_A$ of meta arguments supporting object-level arguments as:

$$\begin{aligned}\mathcal{S}_A = \{(Arg_1, Arg_2) \mid Arg_1 \in \mathcal{A}_M, Arg_2 \in \mathcal{A}_O,\\ (\mathsf{Conc}(Arg_1) = bp\_met(Arg_2) \vee \mathsf{Conc}(Arg_1) = \neg burdened(Arg_2))\}\end{aligned}$$

Intuitively, an argument A at the object level is supported by arguments at the meta level claiming that either the burden on A is satisfied (S2) or there is no burden allocated on it (S1).

The set $\mathcal{S}_R$ of meta arguments supporting object-level attacks is defined as:

$$\begin{aligned}\mathcal{S}_R = \{(Arg_1, (B, A)) \mid Arg_1 \in \mathcal{A}_M, (B, A) \in \mathcal{R}_O,\\ \mathsf{Conc}(Arg_1) = \mathit{effectiveAttack}(B, A)\}\end{aligned}$$

In other words, an object-level attack is supported by arguments at the meta level claiming its effectiveness w.r.t. the burden of persuasion allocation (S4).

Equivalence with burden of persuasion semantics

The defined meta-framework can be used to achieve the same results of the original burden of persuasion labelling semantics [Calegari et al., 2021d].

Let us first introduce the notion of *CQ-consistency* for a bimodal argumentation graph G.

Definition 44 (CQ-consistency). *Let $G = \langle \mathcal{A}_O, \mathcal{A}_M, \mathcal{R}_O, \mathcal{R}_M, \mathcal{S}_A, \mathcal{S}_R \rangle$ be a bimodal argumentation graph, and let $L_S(G)$ be a labelling-based semantics. P is the set of corresponding L_S-perspectives. A perspective $p \in P$ is CQ-consistent if every* IN *argument A in the corresponding meta-level labelling satisfies its critical questions ($CQ(A)$).*

Before proceeding, let us ground the Critical Questions introduced in Subsection 3.5.3 within the context of L_S-perspectives and labelling based semantics.

CQ_{S0} Given a L_S-perspective p and one of its labelling l, is $l(B) = \mathsf{IN}$?

CQ_{S1} Given a L_S-perspective p and one of its labelling l, is $l(A) = \mathsf{IN}$? If an argument B such that $\mathsf{Conc}(B) = bp(\phi)$ does exist, is $l(B) \in \{\mathsf{UND}, \mathsf{OUT}\}$?

CQ_{S2} Given a L_S-perspective p and one of its labelling l, is $l(A) = \mathsf{IN}$?

CQ_{S2} Given all L_S-perspectives p and the set of their labellings L, does $\forall l \in L, l(A) \in \{\mathsf{UND}, \mathsf{OUT}\}$ hold?

CQ_{S3} Given a L_S-perspective p and one of its labelling l, are $l(A) = \mathsf{IN}$ and $l(B) = \mathsf{IN}$?

Using this new definition we can introduce the concept of *BP-perspective.*

Definition 45 (BP-perspective)**.** *Let $G = \langle \mathcal{A}_O, \mathcal{A}_M, \mathcal{R}_O, \mathcal{R}_M, \mathcal{S}_A, \mathcal{S}_R \rangle$ be a bimodal argumentation graph, and P the set of its L_{stable}-perspectives [Baroni et al., 2011c]. We say that $p \in P$ is a BP-perspective of G iff p is CQ-consistent.*

Before proceeding, let us recall the main definitions from Calegari and colleagues [Calegari et al., 2021d], who, in their work, present a semantics dealing with the burden of persuasion allocation on members of the argumentation language.

Definition 46 (BP-defeat [Calegari et al., 2021d])**.** *Given a set of burdens of persuasion $\mathsf{BurdPers}$, A **bp-defeats** B iff there exists a subargument B' of B such that:*

1. *$\mathsf{Conc}(A) = \overline{\mathsf{Conc}(B')}$ and*
   - *(a) $\mathsf{Conc}(A) \notin \mathsf{BurdPers}$, and $B' \not\succ A$, or*
   - *(b) $\mathsf{Conc}(A) \in \mathsf{BurdPers}$ and $A \succ B'$.*
2. *$\mathsf{Conc}(A) = \neg \mathsf{N}(\rho)$, where ρ is $\mathsf{TopRule}(B')$.*

Definition 47 (Grounded BP-labelling [Calegari et al., 2021d])**.** *A grounded **BP-labelling** of an argumentation graph G, relative to a set of burdens* $\mathsf{BurdPers}$*, is a* $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$*-labelling l s.t. the set of* UND *arguments is minimal and* $\forall \mathsf{A} \in \mathcal{A}_G$ *with* $\mathsf{Conc}(A) = \phi$

1. $l(A) = \mathsf{IN}$ *iff* $\forall \mathsf{B} \in \mathcal{A}_G$ *such that* B *bp-defeats* $\mathsf{A} : l(B) = \mathsf{OUT}$

2. $l(A) = \mathsf{OUT}$ *iff*

   (a) $\phi \in \mathsf{BurdPers}$ *and* $\exists\ B \in \mathcal{A}_G$ *s.t. B bp-defeats A and* $l(B) \neq \mathsf{OUT}$

   (b) $\phi \notin \mathsf{BurdPers}$ *and* $\exists\ B \in \mathcal{A}_G$ *such that B bp-defeats A and* $l(B) = \mathsf{IN}$

3. $l(A) = \mathsf{UND}$ *otherwise.*

Proposition 4. *If* $\nexists A, B \in \mathcal{A}_O$ *such that both A and B have a burden of persuasion on them and A is reachable from B through* $\mathcal{R}_O$*, the results yielded by the grounded evaluation of G's BP-perspectives are congruent with the evaluation of the object-level graph* $\langle \mathcal{A}_O, \mathcal{R}_O \rangle$ *under the Grounded BP-labelling as in Definition 47 [Calegari et al., 2021d].*

Proof. The burden of persuasion semantics acts like the grounded semantics, with the only difference that the burdened arguments that would have been UND for the latter are possibly OUT/IN for the former. So, it is a matter of fact that burdened arguments and arguments connected to them through attack relation can change their state.

Let us consider an argumentation graph $AF\langle \mathcal{A}, \rightsquigarrow \rangle$, and let L_G be the grounded labelling resulting from the evaluation of AF under a grounded semantics. With respect to our framework, and in particular, to the bimodal argumentation graph $G = \langle \mathcal{A}_O, \mathcal{A}_M, \mathcal{R}_O, \mathcal{R}_M, \mathcal{S}_A, \mathcal{S}_R \rangle$, we have, by construction, that every node at the object level, if not burdened, has an undisputed supporting argument at the meta level ($\mathsf{S1}$ or $\mathsf{S4}$). As a consequence, the meta level has no influences on no burdened arguments, and – in the absence of burdened arguments – the evaluation of the object level graph under the grounded semantics would be equal to L_G. It is a matter of fact that the meta level influences only the burdened arguments' state. Accordingly, the extent of this influence and the consequences on the object-level graph will be considered in the following.

Let us consider a single argument $A \in \mathcal{A}$ allocated with the burden of persuasion, thus having the additional argument $B \in \mathcal{A}$ stating the burden on A (as depicted in Figure 3.4). Computing the stable semantics on the meta-level graph produces the following scenarios:

Stable.a burden on A cannot be proved;

Stable.b burden on A can be proved and the burden is met;

Stable.c burden on A can be proved and the burden is not met.

Accordingly, the stable evaluation of the meta-graph produces three different perspectives of the object level:

(i) argument A is supported—it is not burdened;

(ii) argument A is supported—it satisfies the burden;

(iii) argument A is not supported, and then it is excluded from the object-level graph—it does not meet the burden then it is refuted.

In particular, we have that Stable.a induces *(i)*, Stable.b leads to *(ii)*, while Stable.c induces *(iii)*. Let L_{BP} be this new object-level labelling (obtained by the meta-level stable semantics reification at the object level). Also, let us compare L_{BP} with the initial object-level grounded labelling L_G. Then, the following cases can occur (E is exploited for valid solutions with labelling equivalence, while C is exploited for solutions to be discarded).

- B is OUT or UND in L_G.

  E1 If *(i)* the burden is not allocated and cannot be proven, the meta level does not influence the object level supporting all unburdened arguments. CQ_{S1} is satisfied and L_{BP} is equivalent to L_G.

  C1 If *(ii)* or *(iii)*, in both cases CQ_{S0} is not satisfied—the burden is proved at the meta level and not at object level.

- B is IN and A is OUT in L_G.

C2 If *(i)* we have an inconsistency on CQ_{S1}—the burden is proved at object level and not at meta level.

C3 If *(ii)* we have an inconsistency on CQ_{S2} since A is considered IN at the meta level (supported by the meta-argument) but A is OUT at the object level.

E2 If *(iii)* A is not supported, i.e., removed from the object-level graph. CQ_{S0} and CQ_{S3} are both satisfied. Then, under the grounded semantics, the removal of an OUT argument from a graph is not influent w.r.t. its evaluation, i.e., L_{BP} is equivalent to L_G.[1]

- B is IN and A is IN in L_G.

C4 If *(i)*, we have an inconsistency on CQ_{S1}—the burden is proved at object level and not at meta level.

E3 If *(ii)*, then CQ_{S0} and CQ_{S2} are both satisfied and L_{BP} is equal to L_G.

C5 If *(iii)* we have an inconsistency because CQ_{S3} is not satisfied.

- B is IN and A is UND in L_G.

C6 If *(i)*, we have an inconsistency on CQ_{S1}—the burden is proved at object level and not at meta level.

C7 If *(ii)*, we have an inconsistency since A is considered IN at the meta level (supported by the meta-argument) but A is UND at the object level—CQ_{S2} is not satisfied.

E4 If *(iii)* A is not supported, i.e., is removed from the object level, i.e., it can be labelled as OUT in L_{BP} (see [1]). CQ_{S0} and CQ_{S2} are satisfied.

As made evident by the proof, the reification of the meta level upon the object level generates multiple solutions: yet, only one solution for each case can be considered valid w.r.t. critical questions. Moreover, the only valid perspective coincides with

[1] It can trivially be proved considering that – in the grounded semantics – an OUT argument does not affect other arguments' state, i.e., it is irrelevant and can be removed; of course, also the dual proposition holds, i.e., if L_{BP} build in the meta-frameworks does not consider an argument it can be labelled as OUT in the grounded bp-labelling

the one generated from the bp-labelling in [Calegari et al., 2021d]—the burdened argument is labelled OUT in case of indecision (E4). Obviously, the proof can be generalised to configurations taking into account any number of burdened independent arguments—where combinations grow exponentially with the number of burdened arguments. □

Example 6 (Antidiscrimination law)**.** *Let us consider a case in which a woman claims to have been discriminated against in her career on the basis of her sex, as she was passed over by male colleagues when promotions came available (ev*1*), and brings evidence showing that in her company all managerial positions are held by men (ev*3*), even though the company's personnel includes many equally qualified women, having worked for a long time in the company, and with equal or better performance (ev*2*). Assume that this practice is deemed to indicate the existence of gender-based discrimination (indiciaDiscrim) and that the employer fails to provide prevailing evidence that the woman was not discriminated against (¬discrim). It seems that it may be concluded that the woman was indeed discriminated against on the basis of her sex.*

Consider, for instance, the following formalisation of the European nondiscrimination law, that, in case of presumed discrimination, requires prevailing evidence that no offence was committed—i.e., bp(¬discrim):

$$\begin{array}{lll}
e1 : ev1 & e2 : ev2 & e3 : ev3 \\
er1 : ev1 \Rightarrow indiciaDiscrim & er2 : ev2 \Rightarrow \neg discrim & er3 : ev3 \Rightarrow discrim \\
r1 : indiciaDiscrim \Rightarrow bp(\neg discrim) & &
\end{array}$$

We can then build the following object-level arguments:

$$\begin{array}{lll}
A_0 :\Rightarrow ev1 & B_0 :\Rightarrow ev2 & C_0 :\Rightarrow ev3 \\
A_1 : A_0 \Rightarrow indiciaDiscrim & B_1 : B_0 \Rightarrow \neg discrim & C_1 : C_0 \Rightarrow discrim \\
A_2 : A_1 \Rightarrow bp(\neg discrim) & &
\end{array}$$

and the following meta-level arguments:

$$\begin{array}{ll}
A_{0_{S1}} :\Rightarrow -\mathit{burdened}(A_0) & B_{0}S1 :\Rightarrow -\mathit{burdened}(B_0) \\
A_{1_{S1}} :\Rightarrow -\mathit{burdened}(A_1) & B_{1_{S0}} :\Rightarrow \mathit{burdened}(B_1) \\
A_{2_{S1}} :\Rightarrow -\mathit{burdened}(A_2) & B_{1_{S1}} :\Rightarrow -\mathit{burdened}(B_1) \\
C_{0_{S1}} :\Rightarrow -\mathit{burdened}(C_0) & B_{1_{S2}} : B_{1_{S0}} \Rightarrow \mathit{bp\_met}(B_1) \\
C_{1_{S1}} :\Rightarrow -\mathit{burdened}(C_1) & B_{1_{S3}} : B_{1_{S0}} \Rightarrow \neg \mathit{bp\_met}(B_1) \\
C_1B_{1_{S4}} :\Rightarrow \mathit{effectiveAttack}(C_1, B_1) & B_1C_{1_{S4}} :\Rightarrow \mathit{effectiveAttack}(B_1, C_1)
\end{array}$$

The resulting graph is depicted in Figure 3.5. In this case, at the object level, since there are indicia of discrimination (A_1), we can infer the allocation of the burden on non-discrimination (A_2). Moreover, we can build both arguments for discrimination (C_1) and non-discrimination (B_1), leading to a situation of undecidability.

At the meta level we can apply the rule S1 for every argument at the object level ($A_{0_{S1}}, A_{1_{S1}}, A_{2_{S1}}, B_{0}S1, B_{1_{S0}}, C_{0_{S1}}, C_{1_{S1}}$) – where we can establish the absence of the burden for all of them –, and the rule S4 for every attack ($C_1B_{1_{S4}}, B_1C_{1_{S4}}$). By exploiting B_1 and A_2, we can also apply schema S0, and consequently rules S2 and S3. In a few words, we are concluding the meta argumentative structure given by the allocation of the burden of persuasion on argument B_1.

We can now apply the stable labelling to the meta-level graph, thus obtaining three distinct results. For clarity reasons, in the following we ignore the arguments that are acceptable under every solution.

1. $\mathsf{IN} = \{B_{1_{S1}}, C_1B_{1_{S4}}, B_1C_{1_{S4}}\}, \mathsf{OUT} = \{B_{1_{S0}}, B_{1_{S2}}, B_{1_{S3}}\}, \mathsf{UND} = \{\}$*—i.e.,* B_1 *is not burdened;*

2. $\mathsf{IN} = \{B_{1_{S0}}, B_{1_{S2}}, C_1B_{1_{S4}}, B_1C_{1_{S4}}\}, \mathsf{OUT} = \{B_{1_{S1}}, B_{1_{S3}}\}, \mathsf{UND} = \{\}$*—i.e.,* B_1 *is burdened and the burden is met;*

3. $\mathsf{IN} = \{B_{1_{S0}}, B_{1_{S3}}\}, \mathsf{OUT} = \{B_{1_{S1}}, B_{1_{S2}}, C_1B_{1_{S4}}, B_1C_{1_{S4}}\}, \mathsf{UND} = \{\}$*—i.e.,* B_1 *is burdened and the burden is not met.*

Then, the meta-level results can be reified to the object-level perspectives taking into account the CQ *we have to impose on the solutions and the results given by the perspective evaluation under the grounded semantics. Let us first consider solutions 1 and 2. They lead to the same perspective on the object-level graph—the graph remains unchanged w.r.t. the original graph. If we consider the critical questions attached to the* IN *arguments, both these solutions are not valid. Indeed,* *according to solution 1 the burden is not allocated on argument B_1, but this is in contrast with argument A_2's conclusion (A_2 is* IN *under grounded labelling)—i.e., CQ_{S1} is not satisfied. Analogously, solution 2 concludes that B_1 is allocated with the burden and its success to meet the burden, but at the same time, argument B_1 is undecided at the object level (B_1 is* UND *under the grounded semantics)—i.e., CQ_{S2} is not satisfied.*

The only acceptable result is the one given by solution 3. In this case, argument B_1 is not capable to meet the burden – $B_{1_{S3}}$ is IN *– and, consequently, it is rejected and deleted from the perspective. Indeed, CQ_{S3} is satisfied. As a consequence, argument C_1 is labelled* IN*. In other words, the argument for non-discrimination fails and the argument for discrimination is accepted.*

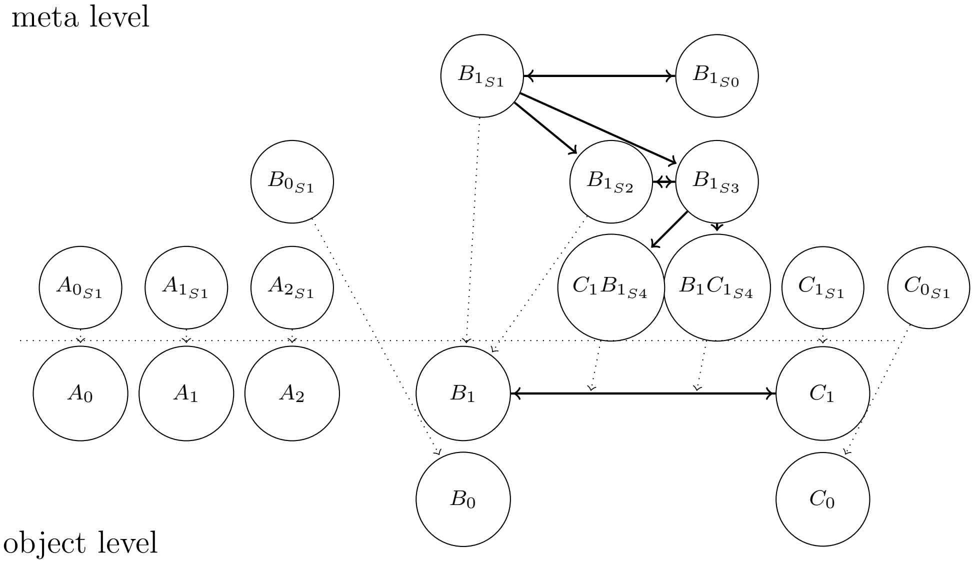


Figure 3.5: Argumentation graph (object- and meta- level) from Example 6

Let us consider a situation in which one argument A is presented for a claim ϕ being burdened, and A (or one of its subarguments) is attacked by a counterargument B, of which the conclusion ψ is also burdened. Intuitively, if both arguments fail to satisfy the burden of persuasion, both of them are to be rejected. This is not the case if the inversion of the burden is taken into account [Calegari et al., 2021d]—i.e., if no convincing argument for ψ is found, then the attack fails, and the uncertainty on ψ does not affect the status of A. Accordingly,

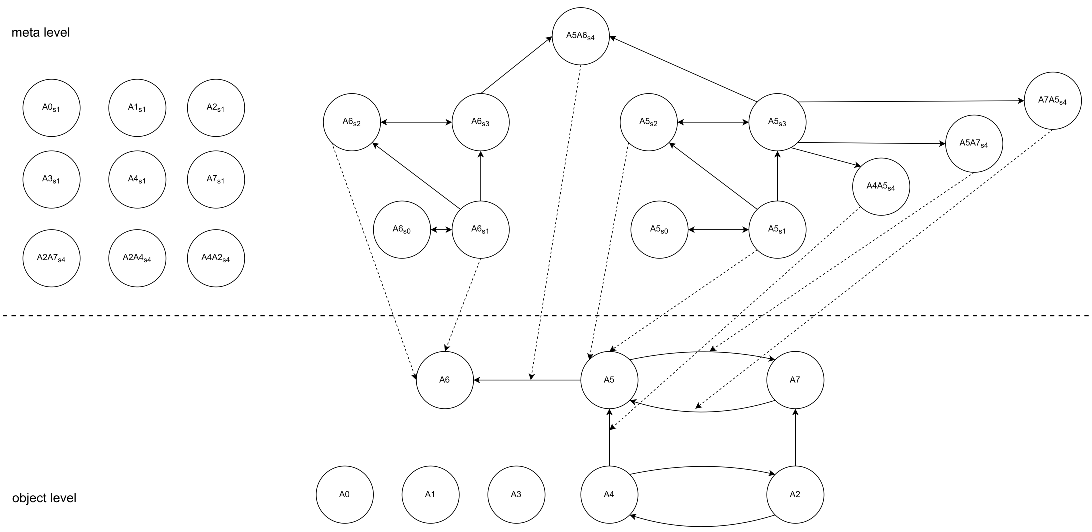


Figure 3.6: Argumentation graph (object- and meta- level) from Example 7

B is rejected for failing to meet its burden, thus leaving A free to be accepted also if it was not able to satisfy the burden of persuasion in the beginning.

The proposed model is able to correctly deal with this exception, as we discuss in the next example adapted from [Calegari et al., 2021d].

Example 7 (Inversion of the burden)**.** *Let us consider a case in which a doctor caused harm to a patient by misdiagnosing his case. Assume that there is no doubt that the doctor harmed the patient (harm), but it is uncertain whether the doctor followed the guidelines governing this case. Assume that, under the applicable law, doctors are liable for any harm suffered by their patients (liable), but they can avoid liability if they show that they exercised due care in treating the patient (dueDiligence). Let also assume that a doctor is considered to be diligent if he/she follows the medical guidelines that govern the case (guidelines). The doctor has to provide a convincing argument that he/she was diligent ($bp(dueDiligence)$), and the patient has to provide a convincing argument for the doctor's liability ($bp(liable)$).*

We can formalise the case as follows:

$$
\begin{array}{ll}
f1 : guidelines & f2 : \neg guidelines \\
f3 : harm & r1 : \neg guidelines \Rightarrow \neg dueDiligence \\
r2 : guidelines \Rightarrow dueDiligence & r3 : harm, \sim dueDiligence \Rightarrow liable \\
bp1 : bp(dueDiligence) & bp2 : bp(liable)
\end{array}
$$

We can then build the following object-level arguments:

$$\begin{array}{ll}
A_0 :\Rightarrow bp(dueDiligence) & A_1 :\Rightarrow bp(liable) \\
A_2 :\Rightarrow guidelines & A_3 :\Rightarrow harm \\
A_4 :\Rightarrow \neg guidelines & A_5 : A_2 \Rightarrow dueDiligence \\
A_6 : A_3 \Rightarrow liable & A_7 : A_4 \Rightarrow \neg dueDiligence
\end{array}$$

According to the original burden semantics, the argument for the doctor's due diligence (A_5) fails to meet its burden of persuasion. Consequently, following the inversion principle, it fails to defeat the argument for the doctor's liability (A_6), which is then able to meet its burden of persuasion.

Let's now analyse the case under the meta-model perspective. Using argument schemes defined in Section 3.5 we can build the following meta-arguments:

$$\begin{array}{ll}
A_{0_{S1}} :\Rightarrow -burdened(A_0) & A_{1_{S1}} :\Rightarrow -burdened(A_1) \\
A_{2_{S1}} :\Rightarrow -burdened(A_2) & A_{3_{S1}} :\Rightarrow -burdened(A_3) \\
A_{4_{S1}} :\Rightarrow -burdened(A_4) & A_{7_{S1}} :\Rightarrow -burdened(A_7) \\
A_2A_{7_{S4}} :\Rightarrow effectiveAttack(A_2, A_7) & A_2A_{4_{S4}} :\Rightarrow effectiveAttack(A_2, A_4) \\
A_4A_{2_{S4}} :\Rightarrow effectiveAttack(A_4, A_2) & \\
A_7A_{5_{S4}} :\Rightarrow effectiveAttack(A_7, A_5) & A_5A_{7_{S4}} :\Rightarrow effectiveAttack(A_5, A_7) \\
A_4A_{5_{S4}} :\Rightarrow effectiveAttack(A_4, A_5) & A_5A_{6_{S4}} :\Rightarrow effectiveAttack(A_5, A_6) \\
A_{5_{S0}} :\Rightarrow burdened(A_5) & A_{5_{S1}} :\Rightarrow -burdened(A_5) \\
A_{5_{S2}} : A_{5_{S0}} \Rightarrow bp\_met(A_5) & A_{5_{S3}} : A_{5_{S0}} \Rightarrow \neg bp\_met(A_5) \\
A_{6_{S0}} :\Rightarrow burdened(A_6) & A_{6_{S1}} :\Rightarrow -burdened(A_6) \\
A_{6_{S2}} : A_{6_{S0}} \Rightarrow bp\_met(A_6) & A_{6_{S3}} : A_{6_{S0}} \Rightarrow \neg bp\_met(A_6)
\end{array}$$

Connecting the object- and meta-level arguments we obtain the graph in Figure 3.6. Let us now consider the extensions obtained applying stable semantics to the meta-level graph:

1. $\{A_{6_{S0}}, A_{6_{S2}}, A_{5_{S0}}, A_{5_{S3}}\}$

2. $\{A_{6_{S0}}, A_{6_{S3}}, A_{5_{S0}}, A_{5_{S3}}\}$

3. $\{A_{6_{S0}}, A_{6_{S2}}, A_{5_{S0}}, A_{5_{S2}}, A_5A_{6_{S4}}, A_5A_{7_{S4}}, A_7A_{5_{S4}}, A_4A_{5_{S4}}\}$

4. $\{A_{6_{S0}}, A_{6_{S3}}, A_{5_{S0}}, A_{5_{S2}}, A_5A_{7_{S4}}, A_7A_{5_{S4}}, A_4A_{5_{S4}}\}$

5. $\{A_{6_{S0}}, A_{6_{S2}}, A_{5_{S1}}, A_5A_{6_{S4}}, A_5A_{7_{S4}}, A_7A_{5_{S4}}, A_4A_{5_{S4}}\}$

6. $\{A_{6_{S0}}, A_{6_{S3}}, A_{5_{S1}}, A_5A_{7_{S4}}, A_7A_{5_{S4}}, A_4A_{5_{S4}}\}$

7. $\{A_{6_{S1}}, A_{5_{S0}}, A_{5_{S2}}, A_5A_{6_{S4}}, A_5A_{7_{S4}}, A_7A_{5_{S4}}, A_4A_{5_{S4}}\}$

8. $\{A_{6_{S1}}, A_{5_{S1}}, A_5A_{6_{S4}}, A_5A_{7_{S4}}, A_7A_{5_{S4}}, A_4A_{5_{S4}}\}$

9. $\{A_{6_{S1}}, A_{5_{S0}}, A_{5_{S3}}\}$

The only extensions that produce a CQ-consistent perspective are the first and the second, given that all the others violate at least one of the constraints imposed by the critical questions—e.g. CQ_{S1} for $5, 6, 7, 8, 9$ and CQ_{S2} for $3, 4$. The first perspective acts exactly like the original semantics from [Calegari et al., 2021d]—i.e., the argument for the doctor's due diligence (A_5) fails to meet the burden ($A_{5_{S3}}$), and consequently, the argument for doctor's liability (A_6) is able to satisfy its own burden ($A_{6_{S2}}$). However, the model delivers a second result according to which both A_5 and A_6 fail to meet their burden of persuasion ($A_{6_{S3}}$ and $A_{5_{S3}}$). It is the result that we would have expected in absence of the inversion principle.

The example shows how the meta-argumentation model is able to provide both a solution that follows the inversion principle and the one not considering it. In general, when the inversion principle is taken into account the number of burdened arguments are maximised in the final extension. Accordingly, we can provide a generalisation of Property 4:

Proposition 5. *Given the results yield by the grounded evaluation of G's BP-perspectives, the results that maximise the number of burdened arguments in the* IN *set are congruent with the evaluation of the object-level graph $\langle \mathcal{A}_O, \mathcal{R}_O \rangle$ under the grounded-bp semantics as in Definition 47 [Calegari et al., 2021d].*

Chapter 4

Optimising the Argumentation Resolution Process

In this chapter, we start to bridge the gap between the formal theoretical results of argumentation and their practical applications.

Argumentation often grapples with problems of high complexity. This complexity is not merely an academic concern; it has tangible implications in the real world where decision-making processes and problem-solving strategies hinge on the efficient and effective resolution of arguments. In practical settings, the intricate web of premises, conclusions, and rebuttals requires not just understanding, but also an efficient approach to manage and resolve.

However, the intricate and often computationally intensive nature of formal argumentation poses a significant challenge. This complexity is heightened by the need to integrate these formal systems into the fabric of everyday technology, which is both ubiquitous and varied. Thus, the primary challenge we face is finding a compromise that allows the rigorous results of argumentation theory to be effectively and efficiently applied in practice.

Moreover, we cannot overlook the role of today's pervasive technology. The ubiquity of computing devices offers a unique platform to implement these algorithms, making the principles of formal argumentation more accessible and applicable than ever before.

As we progress through this chapter, we will explore how the mechanisms of argumentation can be modified to fulfill these requirements, especially by the distribution of the resolution process. Additionally, we will examine a possible approximation of the formal burden of persuasion model that was introduced in the previous chapter. This approximation is not to be intended as a mere simplification; rather, it is a way to maintain the soundness of argumentation principles while adapting it for practical use. The goal is to develop algorithms that are both theoretically sound and practically viable, capable of operating within the constraints of real-world computational environments.

4.1 Structured Reasoning

In the context of structured argumentation, computation is generally performed in two distinct steps. First, all the arguments have to be derived from the argumentation theory; then, the labelling on the resulting argumentation graph can be computed. From the perspective of computational efficiency, the process is highly expensive. The engine performs an exhaustive search on the knowledge base to derive the argumentation trees from facts and rules. Intuitively, the computational cost of the procedure is bound to the number of inference steps to perform—i.e., for every new node in the argumentation tree, the entire rule base has to be examined again to verify the existence of another inference step. Consequently, the cost of the transformation grows both with the rule base dimension and with arguments' articulation, thus making it difficult to use the procedure when a large set of data is available. Of course, correct implementations can help to mitigate the efficiency problems – for instance, exploiting caching or high-performance data structures – but the procedure is inherently inefficient. As for the labelling algorithm, their complexity is a well-known problem in the literature – grounded semantic is the only one having polynomial complexity [Kröll et al., 2017a].

The introduction of a structured reasoning algorithm presents a potential way to enhance efficiency, as it eliminates the need to construct the entire argumentation graph when evaluating a single query. The algorithm delivers – in the average case – a much more efficient way to verify the admissibility of an argument when compared with the standard graph mode. The argumentation graph needs not be entirely derived: instead, it can be explored only as required to verify the state of

Listing 4.1: Structured argumentation, Arg-tuProlog answer query algorithm for grounded semantic (pseudo-code).

```
AnswerQuery(Goal):
  $A_1, ..., A_n$ = buildSustainingArguments(Goal)
  Res = ∅
  for A in $A_1, ..., A_n$:
    Res = Res ∪ Evaluate(A, ∅)
  return Res.

Evaluate(A, Chain):
  if(∃ B ∈ Attacker(A): Evaluate(B, A ∪ Chain) = IN)
    return OUT
  if(∃ B ∈ Attacker(A): B ∈ Chain)
    return UND
  if(∃ B ∈Attacker(A): Evaluate(B, A ∪ Chain) = UND)
    return UND
  return IN.
```

the queried claim. In the case of fully-connected graphs, depending on arguments configuration, it could be necessary to derive all the arguments to have a response, and in these cases – the worst – the algorithm complexity would be the same as that of the standard procedure. Nevertheless, in general, this operation mode potentially avoids the evaluation of whole portions of the argumentation graph, thus hugely increasing the efficiency of the computation.

The algorithm used to evaluate a single claim (or query) according to grounded semantic is inspired by the DeLP dialectical trees evaluation [García and Simari, 2004]. Listing 4.1 shows the pseudo-code – `AnswerQuery(Goal)` – for the `answerQuery/4` predicate: given a claim (`Goal`) as input, the function first builds all the arguments sustaining that claim (`buildSustainingArguments(Goal)`), and then requires their evaluation via the `Evaluate(A, Chain)` function. In order to assess the $A_1, ..., A_n$ status (acceptability or rejection), three conditions are evaluated:

(Cond1) if a conflicting argument labelled as IN exists, then A_1 is OUT;

(Cond2) if a cycle in the route from the root to the leaves (`Chain`) exists, then A_1 argument is UND;

(Cond3) if a conflicting argument labelled as UND exists, then also the A_1 argument is UND.

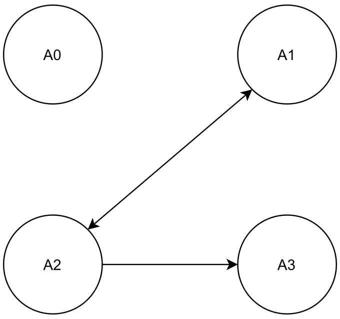


Figure 4.1: Argumentation graph for arguments from Example 8, in which nodes are arguments and edges are attacks between arguments.

If none of the above conditions is met, then the argument can be accepted. Indeed, the states on which such conditions are not met are only two: there are no conflicting arguments – the argument is a leaf in the dialectical tree – or the conflicting arguments are all OUT. All arguments $A_2, ..., A_n$ are then evaluated repeating the same procedure.

Example 8. *Let us consider the following argumentation theory and the corresponding arguments (also depicted in Figure 4.1).*

$$\begin{array}{llllllll}
r1 & : & & \Rightarrow a & \qquad A0 & : & & \Rightarrow_{r1} a \\
r2 & : & a & \Rightarrow b & \qquad A1 & : & A0 & \Rightarrow_{r2} b \\
r3 & : & & \Rightarrow \neg b & \qquad A2 & : & & \Rightarrow_{r3} \neg b \\
r4 & : & b & \Rightarrow c & \qquad A3 & : & A1 & \Rightarrow_{r4} c
\end{array}$$

According to grounded semantic $A0$ *is* IN *– there are no arguments contending its claim or undercutting its inferences – while* $A1$, $A2$ *and* $A3$ *are* UND*—*$A1$ *and* $A2$ *have opposite conclusions and thus attack each other; the conflict is then propagated to the derived argument* $A3$.

Let us suppose we require the evaluation of claim b through the `AnswerQuery(Goal)` *function in Listing 4.1. First, the arguments sustaining b are created, in this case only* $A1$. *Then the evaluation conditions on* $A1$ *attackers – only* $A2$ *in this case – are assessed. However,* $A2$ *admissibility depends, in turn, on* $A1$*—as you can see in Figure 4.1 also* $A1$ *attacks* $A2$. *There is a cycle in the graph (Cond2), and no other attackers matching (Cond1). As a consequence,* $A2$ *is* UND *and thus* $A1$ *(Cond3). Accordingly, claim b is labelled* UND *as expected.*

4.2 Distributed Reasoning

In this section, we shift our focus on how to effectively distribute its argumentation process (evaluation of arguments) so as to enable the exploitation of argumentation techniques in the context of cooperative argumentation.

A first version of a message-based distributed argumentation algorithm is here discussed as the basic pillar of a computational model for cooperative argumentation in MAS. We ignore issues such as agent autonomy and MAS coordination artefacts [Oliva et al., 2008, Oliva et al., 2009], and focus instead on the distribution issues of cooperative argumentation, which enables agent dialogue and defeasible reasoning in MAS.

The first issue when facing computational issues of cooperative argumentation is the parallelisation of the argumentation process. Parallelisation needs to be tackled under two distinct perspective: *(i)* the algorithmic perspective and *(ii)* the data perspective. Under the algorithmic perspective, we try to divide the argument evaluation (w.r.t. a given semantics) into smaller sub-tasks to be executed in parallel. Under the data perspective, instead, we try to achieve parallelisation by splitting the data used by the algorithm—i.e., the argumentation defeasible theory. Action here is therefore at the data level, looking for possible data partitioning on which the argumentation process can be run in parallel.

Before going into more details, we introduce the algorithm that served as a starting point in the parallelisation of the argumentation process. Single-query evaluation – as introduced in the last section – is precisely the algorithm we are interested in, given that cooperative argumentation in highly-reactive systems is often based on a quick debate on some beliefs – those concerning the decision to be made at that moment – rather than on a complete assessment of all the agents' knowledge—where a shared agreement is not easily achieved.

Let us now consider the algorithm in Listing 4.1 to analyse the requirements and implications of its parallelisation. The algorithm structure is simple: the argument evaluation leverages the evaluation obtained from its attackers—i.e., the attackers are recursively evaluated using the same algorithm and the result is exploited to determine the state of the target argument. Intuitively, a first point of parallelisation can be found in the search and evaluation of the `Attackers`. Indeed, every condition exploited by the algorithm – (Cond1), (Cond2), and (Cond3) – to evaluate an argument requires one and only one attacker to match the constraint. Those conditions directly suggest an OR parallelisation in the search and evaluation of the attackers. We could evaluate the arguments simultaneously under different branches, and the success in one of the branches would lead to the success of the entire search.

However, the algorithm exposes another point of parallalisation. The order in the evaluation of the conditions is essential for the soundness of the algorithm—as illustrated by the following example.

Example 9. *Let us consider argument A and its two attackers B and C. Let it be the case in which we know B and C's labelling,* IN *for the former and* UND *for the latter. If we do not respect the order dictated by the algorithm, A's labelling is either* UND *(Cond3) or* OUT *(Cond1). Of course, the first result would be in contrast with the original grounded semantic requirements for which every argument having an* IN *attacker should be definitively* OUT*. Conversely, if we respect the evaluation order, A's labelling would be* OUT *in every scenario.*

Although the evaluation order is strict, we can evaluate all the conditions simultaneously and consider the ordering only while providing the labelling for the target argument (mixing AND and OR parallelisation). In other words, the three conditions are evaluated in parallel, but the result is given accordingly to the defined priorities. If (Cond1) is met, the argument is labelled as OUT. Conversely, even if (Cond2) or (Cond3) are met, one should first verify that (Cond1) does not hold. Only then the argument can be labelled as UND.

Listing 4.2 contains the version of the algorithm taking into account both points of parallelisation. The three conditions – (Cond1), (Cond2) and (Cond3) – are evaluated at the same time. Then the results of the three sub-tasks are combined to provide the final solution according to the conditions' priority. Of course, if we consider a scenario where only the first condition (Cond1) is required to determine the status of the argument in input, the parallel evaluation of all the three conditions would lead to a waste of computational resources. However, this problem is easily mitigated by evaluating the sub-task results as soon as they are individually available—i.e. in the case we receive a positive result from a single sub-task,

Listing 4.2: `Evaluate` predicate with both parallel conditions evaluation and parallel attackers

```
Evaluate(A, Chain):
  PARALLEL {
    Cond1 = PARALLEL { ∃ B ∈ Attacker(A): Evaluate(B, A ∪ Chain) =
        IN }
    Cond2 = PARALLEL { ∃ B ∈ Attacker(A): B ∈ Chain }
    Cond3 = PARALLEL { ∃ B ∈ Attacker(A): Evaluate(B, A ∪ Chain) =
        UND }
  }
  if(Cond1) return OUT
  if(Cond2 AND NOT Cond1) return UND
  if(Cond3 AND NOT Cond1) return UND
  if(NOT Cond1 AND NOT Cond2 AND NOT Cond3) return IN
```

and it is enough to compute the argument status, we can cut the superfluous computational branches and return the final solution.

In the first part of our analysis we focused on the parallelisation problem from a pure computational perspective, by discussing whether the evaluation task could be split into a group of sub-task to be executed simultaneously. However, there is another perspective to take into account when parallelising: the one concerning the data.

Example 10. *For example, let us consider a job computing the sum and the product of a set of numbers. Using the sub-task approach, we could have two subroutines running in parallel, one computing the sum and the other computing the product of the numbers. However, leveraging the associativity property of sum and multiplication, we can split the problem into a series of tasks computing both sum and product on a subset of the original data. Then the final result would be the sum and the multiplication of the tasks' results.*

Let us suppose to apply the same principle to the argumentation task. We build arguments from a base theory according to the relations illustrated in Section 2.1. The logic theory is, for all intents, the input data of our algorithm (argumentation task). Now, the question is whether we can effectively split the data into sub-portions to be evaluated in parallel without affecting the global soundness of the original algorithm. Let us consider a splitting principle based on rules dependency

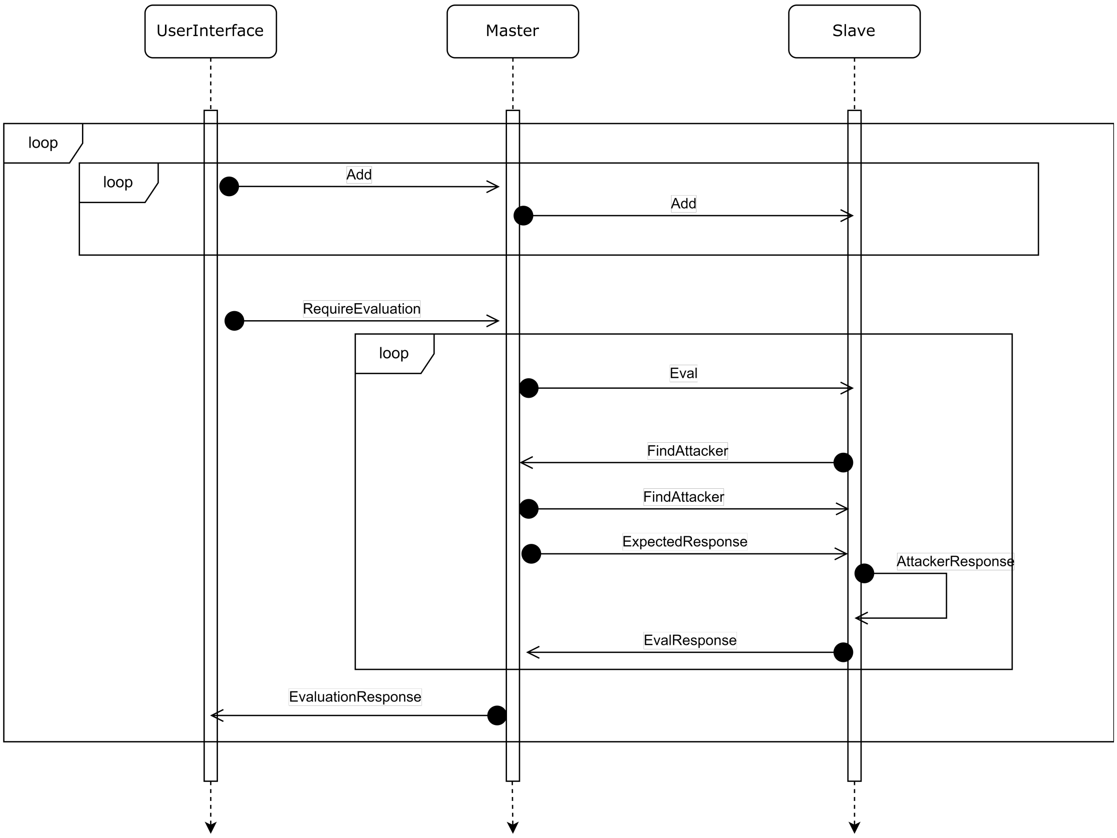


Figure 4.2: Master-slave interaction for argument evaluation.

– i.e., if two rules can be chained, they must stay together –, and the algorithm in Listing 4.2. According to the algorithm, the search and evaluation of the attackers are performed in a distinct subtask (concurrent evaluation). Then, we can split the knowledge concerning attacked and attackers into separate sets, since the subtasks evaluating an attacker require only the knowledge to infer such an attacker—i.e., the *Dependency* principle must be respected. Indeed, there is no task that needs to know how to build both an argument and its attackers, since the search is delegated to another process. In other words, a single subprocess in charge of evaluating an argument needs only the portion of the theory needed to infer the argument itself—i.e., the chainable rules concluding the target claim.

4.2.1 The master-slave actor model

We can now provide a complete and sound mechanism for the admissibility task in a fully-concurrent way, exploiting the insights from Section 4.2 and applying them to an actor-based model [Hewitt et al., 1973].

In short, the actor model is based on a set of computational entities – the actors – communicating each other through messages. The interaction between actors is the key to computation. Actors are pure reactive entities that only in response to a message can:

- create new actors;
- send messages to other actors;
- change their internal state through a predefined behaviour.

Actors work in a fully-concurrent way – asynchronous communication and message passing are fundamental to this end – making the actor model suited to concurrent applications and scenarios. We choose this model for its simplicity: it presents very few abstractions making it easy to study both how to model a concurrent system and its properties. The final goal is to provide a sound model for agents' cooperative argumentation in MAS, enabling concurrent evaluation of the argumentation algorithms (focusing on distribution). The actor paradigm is a straightforward choice for an analysis of this sort.

Since the actor model focuses on actors and their communication, the following design will review the structure and behaviour of the actors involved. Although a fully-distributed version of the model is possible, we choose to adopt a master-slave approach in order to simplify the functioning of the system as much as possible. Accordingly, two main sorts of actors are conceived in the system: *master* and *worker*. Master actors coordinate the knowledge-base distribution phase, while the workers hold a portion of the theory, concurring to the evaluation of a claim through their interaction.

Let us start from the knowledge-base distribution. Since actors are reactive entities, in order to completely adhere to the actor model the master knowledge base can be changed from outside the actor system. If the master receives the order to add a new element to the theory, three possible scenarios can be configured:

1. none of the workers contains a compatible knowledge base – i.e., it is not possible to chain the new rule to the knowledge base – and consequently, the master creates a new worker containing the portion of the theory;

2. one or more workers have a compatible knowledge base, and they add the element to their kb;

3. a set of workers possess overlapping knowledge bases – i.e. the union set of workers' knowledge bases can be used to create a unique inference chain –, and as a consequence, we merge their knowledge bases and destroy the extra workers;

Iterating this procedure for all the elements of an input knowledge base, as a result we should obtain a set of workers each of them containing a portion of the theory in accordance with the *dependency* splitting principle.

Once the knowledge has been correctly split between workers, we can proceed with the actor-based evaluation of an argument. Each actor is responsible for evaluating those arguments that can be build using its portion of the theory. When the actor receives an evaluation request, it first checks if attackers exist, w.r.t. its portion of the knowledge base. Then the actor can: *(i)* register the impossibility to evaluate the argument – only if a cycle through the evaluation chain is detected –, *(ii)* require the attacker arguments evaluation to all the other actors. In the latter case, the actor shall answer the original evaluation request only after receiving a response from others actors. The conditions to match while evaluating an argument are the same as the original algorithm in Listing 4.1:

- if one counterargument is found admissible, we evaluate the argument as `OUT`;

- if any number of actors decide for the argument undecidability with none advancing its rejection, we mark the argument as `UND`;

- if all the actors agree that no counterarguments can be provided as acceptable, we evaluate the argument as `IN`;

Actors provide their suggestions on the state of the requested argument according to all the labels of their counterarguments.

We can describe the interactions between system's actors through a sequence diagram. Messages that masters and workers can exchange are represented in Figure 4.2 with:

- `Add`, sent from the master to a worker, through which the master sends the new theory member to be stored in the workers' kb. The decision on which is the right worker to send the data to is responsibility of the master that knows the entire state of the system and how data has been divided;

- `RequireEvaluation`, sent from outside the system to master to require the evaluation of a claim;

- `Eval`, sent from master to all workers to require the evaluation of a claim;

- `FindAttacker`, sent from a worker to master to require the broadcasting of a request for counterarguments to all the available workers;

- `ExpectedResponses`, sent from master to a worker to communicate the number of expected responses to a request for counterarguments;

- `AttackerResponse`, sent from a worker to a worker in response to a request for counterarguments. The message contains the state of the counterargument obtained through a new `FindAttacker` evaluation;

- `EvalResponse`, sent from workers to master to communicate their decision on a claim. The decision is taken after all the `AttackerResponse` containing the state of possible counterarguments have been received.

- `EvaluationResponse`, message sent from master containing the system decision on the state of a claim;

Note that the `Add` and `RequireEvaluation` messages come from outside the actor system and starts the distribution and evaluation process. This interaction model implement both the parallelization strategies described in Listing 4.2: the search for counterarguments is executed concurrently by all the worker nodes, as also the evaluation on the admissibility of arguments.

Example 11. *Let us consider again the theory in Example 8. Let us assume a single* `MasterActor` *and the following order in the inclusion of the rules in the system: r1, r3, r4, r2.*[1] *As for the first three rules, the behaviour is the same. Since the rules are not chainable, it creates three distinct workers and sends a single rule to every one of them via the* `Add` *message. We now have* `Worker 1`*,* `Worker 2`*, and* `Worker 3` *with respectively r1, r3 and r4 in their knowledge bases. Then the inclusion of rule r2 is required, and both workers 1 and 3 result having a chainable knowledge base. Rule r2 is, in fact, the missing link in the inference chain of r1 and r4. As a consequence, the* `Master` *stops the two workers, creates a new one, and then requires to it the inclusion of rules r1, r4 and r2 via three* `Add` *messages. At the end of the distribution phase, we have two workers, one containing r1, r2, r4, the other just r3. The dependency principle is thus respected. Let us continue the example and require the evaluation of claim b via the* `RequireEvaluation` *message. Consequently, the* `Master` *sends an* `Eval` *message to all the actors.* `Worker 1` *succeeds in building an argument (A1) and sends to all the other* `Workers` *– also* `Worker 1` *is included in the list – a* `FindAttacker` *message requiring attackers evaluation—the broadcasting of the message is done by the* `Master` *actor. The master also communicates the number of responses that are expected (*`ExpectedResponses` *message)—only two in that case.* `Worker 1` *answers with a* `AttackerResponse` *communicating that there are no attacking arguments according to its knowledge, while* `Worker 2` *sends back a* `AttackerResponse` *containing an* `Und` *result. Indeed,* `Worker 2` *is able to create a valid counterargument (A2), but a cycle is detected in the inference chain. According to the evaluation algorithm, receiving an* `Und` *response,* `Worker 1` *can finally label A1 as* UND *and communicate it to the master via a* `EvalResponse` *message.*

4.3 Meta Burden of Persuasion

Despite the benefits of the meta-approach discussed in Section 3.5 – such as clear separation of concerns, encapsulations of argumentation abstractions and naturalness in terms of human thinking – the method is quite inefficient from a compu-

[1] The order of inclusion affects the steps required to converge, not the final state of the system.

tational perspective. Indeed, the meta-level evaluation leads to a stable semantics computation, with a non-polynomial complexity [Kröll et al., 2017b]. This is why, from a technological perspective, the model presented in Section 3.5 has been reified into a more efficient resolution method.

In a nutshell, the proposed approach exploits the stable semantics to explore the search space at the meta level. Then, in order to identify the final solution, the grounded assessment of the object level is taken into account—selecting the acceptable scenario according to the critical questions. The idea behind the technological refinement is exactly to leverage the information of those arguments to guide the search—i.e., to exploit the grounded assessment of the object level as an *a priori* constraint. Following this idea, the computation algorithm becomes really simple. The two argumentation levels (object and meta) are collapsed in a single graph, following [Boella et al., 2009a]. Then, the graph is modified dynamically, leveraging the information on the burdened arguments. In a sense, we have a multi-stage evaluation that leads to the modification of the graph itself at every stage.

Let us consider the framework of Example 6. There two arguments exist, namely B_1 and C_1, attacking each other. Then, another argument, A_2, concludes the presence of the burden on B_1. The grounded evaluation of this framework would lead to a single extension containing argument A_2—i.e., the burden on B_1 has been proved, and we should proceed to verify B_1's compliance with the constraint. According to the model presented in Section 3.5, the graph should be used to build the meta-level framework expressing all the possible outcomes the burden could lead to. Then, the one leading to an object-level perspective that satisfies all the attached *Critical Questions* would be the correct one. This kind of assessment has one major drawback: we already know from the initial grounded evaluation that B_1 does not satisfy its burden; however, through stable semantics, we explore also the scenarios in which B_1's burden is satisfied, just to discard them later using the *Critical Questions*. The main idea of the technological reification presented in this Section is exactly to use the information generated by the initial grounded assessment to produce a new graph including all the new meta-knowledge.

Let us test this approach with the theory in Example 6. We know that B_1 has a burden on it, but it has not been able to satisfy it. As in the original model, we can use this info to build the argument $B_{1_{S3}}$ using the scheme S3. Intuitively, this new argument claims that "B_1 should be rejected for not being able to defend itself" and, consequently, it throws a new attack against it. If we add these new elements to the original framework, we obtain a new framework containing both object- and meta-arguments on the same level. Its evaluation under grounded semantics leads to the expected result: B_1 is rejected, while C_1 and $B_{1_{S3}}$ are both accepted.

More generally, what we are doing is to verify the *Critical Questions* associated with a meta scheme using the grounded evaluation of the original framework. In this way, we do not need stable semantics to explore all the possible scenarios, but, instead, we can directly select the correct one. For instance, in the case of Example 6, $B_{1_{S3}}$ satisfies its critical questions, while $B_{1_{S2}}$ does not. In the case B_1 were able to satisfy its burden, then just $B_{1_{S2}}$ would have been instantiated, and consequently no new attacks would have been introduced in the framework.

Summing up, given a constraint $bp(x)$, then for every argument A having x as its conclusion a new argument B can be introduced in the graph. This argument represents the possibility of A failing/succeeding to meet the burden—expressed by S3 and S2 in the meta-model. A and B's interaction is decided according to the A's ability to satisfy the burden under the grounded semantics:

i) iff A is OUT or UND, then B is an instance of scheme S3, and consequently an attack from B to A is introduced;

ii) iff A is IN, then B is an instance of scheme S2, then no attack is introduced.

Basically, through the first evaluation of the graph, the knowledge required to choose between schemes S3 and S2 is obtained—i.e. the stable semantics evaluation becomes superfluous.

Let us now apply the new approach to Example 7 to see whether the inversion principle is supported or not. If we consider the grounded evaluation of the object-level framework, we obtain two burdened arguments, A_5 and A_6, both failing to satisfy the persuasion constraint. According to our algorithm, we can introduce two meta-arguments based on scheme S3 in the framework, one attacking A_5, and the other A_6. The evaluation of this framework under grounded semantics would of course lead to an undesirable result—i.e. both arguments A_5 and A_6 are rejected.

The enforcement of the inversion mechanism requires a procedural evaluation of the burdened arguments—i.e., we should first evaluate those arguments whose acceptability does not depend on burdened arguments not yet evaluated, and then we apply the algorithm again until all the burdens have been evaluated. For instance, in Example 7 we should first introduce argument $A_{5_{S3}}$ in the graph, and then use the results of this new framework to evaluate the consequences on A_6. Accordingly, the dependencies among burdened arguments are respected—i.e., we enforce the inversion principle.

More formally, given an argumentation framework $AF = \langle \mathcal{A}, \rightsquigarrow \rangle$ along with its grounded extension E_G, we can define the set of burdens to evaluate B_e as

$$\begin{aligned}\{A_0 \in E_G | \mathsf{Conc}(A_0) = bp(a) \ and \ \nexists b \in E_G \ s.t. \\ \mathsf{Conc}(b) = bp\_met(A_1) \ or \ \neg bp\_met(A_1) \ with \ Conc(A_1) = a\}\end{aligned}$$

Then we can define the reduction R_{B_e} of B_e as:

$$\{bp(a) \in B_e \mid \nexists bp(b) \in B_e \ s.t. \ a \ is \ reachable \ from \ b \ through \rightsquigarrow\}$$

In simpler terms, the reduction set contains all the burdens on the arguments whose status does not depend on other burdened arguments. Then, given an AF and its grounded extension we can use the reduction set to produce a new framework AF_1 containing the meta-arguments for the burdens in the set. We can then recursively apply the same procedure on AF_1 until no elements remain to be evaluated in the reduction set. Understandably, the procedure requires the absence of cycles in the burdened arguments in order to derive a partial ordering over the burdens to evaluate. When all the elements in B_e are independent, the reduction set R_{B_e} is the same as B_e—i.e., the procedure is a generalisation of the naive algorithm introduced at the beginning of this section and used in the evaluation of Example 6.

Figure 4.3 shows Example 7's evaluation steps. The graph on the left is obtained from the initial theory. We can compute the set of burdens ($\{A_0, A_1\}$) and

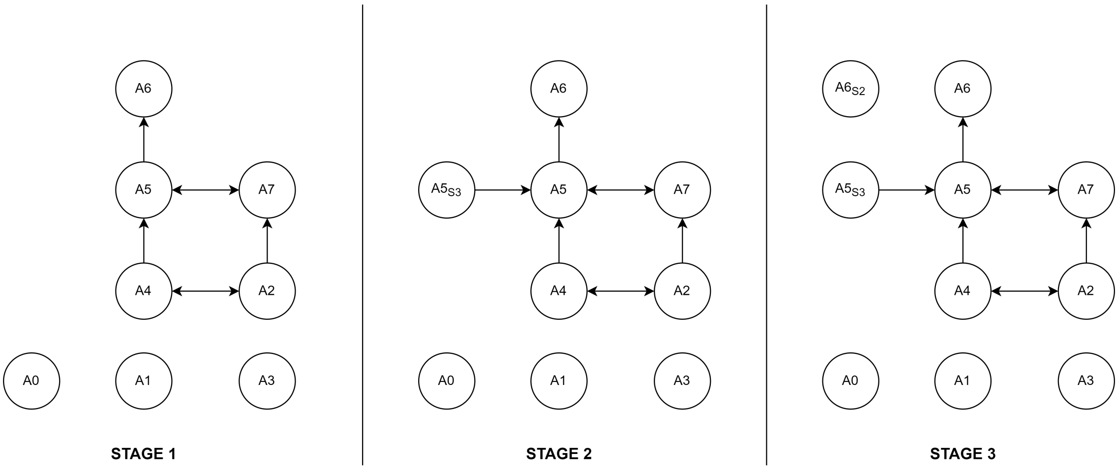


Figure 4.3: Staged evaluation of Example 7

its reduction ($\{A_0\}$). The new knowledge is used to build the framework in the middle by adding an instance of scheme S3 relative to argument A_5 and its attack. Again, we compute the set of burdens ($\{A_0\}$) and its reduction ($\{A_0\}$), and use it to instantiate scheme S2 in the graph on the right. Now the set of burdens to evaluate is empty and we have our final result: argument A_5 fails to satisfy its burden and it is rejected, thus making it possible for A_6 to satisfy its burden.

Chapter 5

Towards a General Argumentation Technology: Arg2P

In this chapter, we transition from the theoretical underpinnings and algorithmic advancements discussed in previous sections to the practical application of these concepts in contemporary technology. This shift marks a crucial step in our exploration, as we now focus on how the refined models and improved algorithms can be effectively integrated into real-world technological systems.

The landscape of intelligent systems today is increasingly defined by a blend of human and artificial agents, computational and physical artifacts, as well as the institutions and norms that govern their interactions. These socio-technical systems demand sophisticated social and organizational concepts and techniques, often derived from the fields of agent and multi-agent systems (MAS) [Omicini and Mariani, 2013]. Particularly, agreement technologies [Ossowski, 2012], play a pivotal role in facilitating intelligent interactions among autonomous agents, fostering cooperation and collaborative activities through dialogue, negotiation, and argumentation.

As we delve deeper into the realm of human-centered intelligent systems, we encounter environments densely populated with agents—both software and human—capable of understanding, arguing, and reporting through factual assertions and arguments [Krippendorff, 2004]. A multi-agent system grounded in argumentation, dialogue, and social judgment becomes an effective foundation for designing systems that are responsive and adaptable to human needs. In such systems, cooperative argumentation is a key strategy for resolving conflicts and reaching consensus [Carrera and Iglesias, 2015].

The role of logic-based technologies in this context is significant, especially in facilitating interaction between humans and agents, and among agents themselves. These technologies offer a framework for defeasible reasoning and agent conversation, with argumentation playing a central role [Andrighetto et al., 2013, Vasconcelos et al., 2002]. In the legal context, for instance, where agents are required not just to reach conclusions but also to explain their reasoning processes, argumentation is particularly relevant [Calegari et al., 2021b]. However, speaking of logic-based technologies can be tricky here: to the best of our knowledge, a logic-based technically-mature environment for argumentation in intelligent systems – both agent-based and accounting for legal aspects – is not available today [Calegari et al., 2019].

For argumentation tools to be able to meet the aforementioned expectations, a considerable effort is required from a software engineering perspective. Indeed, the last decades' continuous improvement in the design and development of technologies for human-centred intelligent systems has not been matched by an analogous improvement of argumentation technologies, where the technological landscape is nowadays populated by very few systems—and most of them are mere prototypes [Calegari et al., 2019]. A key problem in existing argumentation technology is that a widely-acknowledged well-founded computational model for argumentation is currently missing: this makes it difficult to investigate the convergence and scalability of argumentation techniques in highly-distributed environments [Jung et al., 2001, Carrera and Iglesias, 2015]. At the same time, the field has seen a constant flow of a theoretical contributions [Hulstijn and van der Torre, 2004, Modgil and Caminada, 2009].

This chapter addresses this gap by presenting the Arg2P[1] framework, which aims to invigorate the field of argumentation technologies. Designed to be neutral with respect to the underlying argumentation theory, the framework offers a flexible environment where new theoretical contributions can be seamlessly integrated

[1] http://arg2p.apice.unibo.it

and extended.

In order to fit the most advanced application scenarios for intelligent systems – such as pervasive intelligent systems, or the Internet of Intelligent Things (IoIT) [Arsénio et al., 2014] – the Arg2P framework is specifically designed according to the definition of *micro-intelligence* [Omicini and Calegari, 2019, Calegari, 2018], whose key features are *(i)* customisation of the inference methods to be exploited opportunistically in an integrated and easily-interchangeable way, *(ii)* situatedness – i.e., the awareness and reactiveness to the surrounding environment, such as the normative context – and, *(iii)* ability to act at the system micro-level, so as to be easily injectable in disparate contexts and architectures.

Moreover, based on the notion of *ecosystem* of logic-based mechanisms as provided by its technical foundation (tuProlog/2P-Kt [Ciatto et al., 2020]), the design of Arg2P as a logic-based technology offers a huge advantage, where logic programming (LP) itself can become the joining link for diverse extensions of logic (as deduction, abduction, argumentation, just to name a few) while ensuring conceptual integrity of the whole framework. Accordingly, it is designed and developed so as to meet the requirements of observability, interpretability, explicability, accountability, and trustability. Given the requirement of easy integration with existing AI techniques, the technological aspect is of paramount importance in Arg2P, leading to the selection of highly-interoperable languages for the implementation. Also, Arg2P is based on a modular architecture allowing for system openness and ease of extension. Finally, the framework is implemented according to current development practices of continuous integration and continuous delivery (CI/CD). Overall, Arg-tuProlog advances the state of the art both by offering novel implementation of legal concepts fully integrated into an inference and argumentation engine – such as the burdens of proof – and by providing a ready-to-use general purpose argumentation technology for AI applications based on current software engineering standard practices.

Before diving deeper, let's rewind and explore the foundational requirements and the design process that steered the course of the development.

5.1 Components & Requirements

In Section 2.1 we have seen one of the possible formal definitions for a structured argumentation framework. During the years different ways of building an abstract argumentation framework starting from a logic theory have been proposed—e.g. ASPIC+ [Modgil and Prakken, 2014a], ABA+ [Toni, 2014]. Moreover, many extensions to abstract argumentation frameworks have been advanced—e.g. bipolar argumentation frameworks [Amgoud et al., 2004], that deal also with the notion of *support*, or value-based argumentation frameworks [Bench-Capon, 2002], introducing the notion of *value* to determine the defeat relation over arguments. The richness of the theoretical argumentation landscape requires corresponding technologies the ability to deal with all the different nuances that the process could require. Accordingly, the first step for the technology design is the identification of all the common elements of these models. Those will serve as the main requirements and first-class entities of the technology final architecture.

As shown in Section 2.1, the argumentation process can be summarised in the following steps:

1. *definition of the argumentation language*—i.e., elements allowed in rules;
2. *definition of the rules structure*—i.e., how the formulae from the language can be used to build rules;
3. *arguments extraction*—i.e., how rules can be used to build arguments;
4. *attacks extraction*—i.e., how to build the attack relation over the extracted arguments;
5. *arguments evaluation*—i.e., which semantics could be applied to arguments to obtain the framework results;
6. *statements evaluation*—i.e., define how the state of an argument relates to the statement that is being claimed.

We so obtain a finite process that starting from a language leads to the admissibility status of the elements in that language. The most relevant feature of the described process is that it is completely generic, and it may be suitable for a broad range of different models or extensions in the literature.

Let us consider for example the value-based argumentation framework [Bench-Capon, 2002], where every argument is associated with a *value*. A preference relation over the values can be defined and exploited to build the set of valid attacks (*defeat*)—i.e. an argument $A1$ can attack $A2$ only if the value associated with $A2$ is not preferred to $A1$'s one. We can clearly see how all the steps we define above apply also to this special case. More important, we can see that only a limited number of these steps would be impacted by the change w.r.t. the generic procedure we defined:

- arguments extraction, we should now be able to also assign a value to the constructed arguments;
- attacks extraction, the definition of the attack set should depend upon the preference relation over arguments' value.

The rest of the procedure would remain exactly the same. Even though the example above refers to a very specific type of argumentation framework, it can be generalised to a broader set of models and variations sharing the general structure of the argumentation process.

Accordingly, the main entities involved in this process are:

- *Rules*, the basic piece of information in the argumentation language representing an inference from a set of assumptions to a conclusion;
- *Graphs*, the result of all the possible inferences allowed by the rules. Every node in the graph represents a single chain of inferences – *Argument* –, and edges stand for conflicts between arguments—*Attack*;
- *Labelled Graphs*, the result of the evaluation of a graph under a semantics. Every node in the graph is associated with a label giving its final status;
- *Labelled Statements*, the results of the evaluation of a graph under a semantics w.r.t. the claim that arguments advance.

Now, let us introduce the concept of *argumentation pipeline* that will serve as architectural reference in the construction of the Arg2P framework.

Figure 5.1: The argumentation pipeline.

5.1.1 The argumentation pipeline

Figure 5.1 shows the argumentation pipeline that combines the main entities of the architecture above discussed. It contains five main building blocks: *Rule Extractor*, *Arguments Builder*, *Graph Transformer*, *Argument Labeller*, *Statements Labeller.*

Rule Extractor. The first block takes as input a textual source and extracts the corresponding argumentation theory in terms of *Rules* understandable by other blocks in the pipeline. This block encapsulates a way of abstracting from the chosen argumentation language, introducing one more decoupling mechanism. Indeed, all the blocks in the pipeline have to agree on a common interface and structure for the primary entities that we identified above. At the same time, we do not want the final users to be constrained with a predefined language. This is why a *translation* block – the *Rule Extractor* – has been added at the beginning of the pipeline, enabling users to change the input language with no need for modifications in other blocks.

Arguments Builder. The second block in the pipeline is responsible for building arguments and attacks composing the argumentation graph given a set of argumentation rules. This block – the *Arguments Builder* – introduces discretion in the implementation or model kind to be used while building the graph.

Graph Transformer. Let us consider the value-based argumentation framework [Bench-Capon, 2002], where every argument is associated with a *value*. A preference relation over the values can be defined and exploited to build the set of valid attacks (*defeat*)—i.e. an argument $A1$ can attack $A2$ only if the value associated with $A2$ is not preferred to $A1$'s one. In the case at hand, the main difference with a standard abstract argumentation framework is in the value associated with every argument and the use of preference between values to build the attack set. Hence, a mapping from an abstract to a value-based based argumentation framework can be defined according to the following steps:

- add a value to every argument in the input framework;
- discard from the attack set those attacks against a preference on values—e.g., if there is an attack going from A_1 to A_2, but A_2's value is preferred to A_1's one, then discard the attack;

This is the main idea behind the *Graph Transformer* block: to provide an easy way to define and apply transformations in order to introduce (or remove) information from the argumentation graph (thus exploring different application scenarios). More than one transformation can be applied to the graph so to mix their effects in a single framework.

Argument Labeller. The block is responsible for applying a labelling semantics to the input graph and providing a *label* to every argument in it. Semantics is one of the most important things in the definition of a model, and it is possible to find many of them in literature—all giving a different perspective on the input graph. The *Argument Labeller* block is then a generalisation of all the possible different semantics: just changing the block or creating different versions of it enables the output of previous blocks to assume a completely different *meaning*.

Statements Labeller. There could be the case – as usually in structured argumentation – where every argument in a framework stands for a single statement in the input argumentation language. In this case, it is necessary to define how the evaluation of arguments relates to the one of the statements they are claiming. Let us consider the following example in which exists two different arguments, A_1 and A_2, both claiming the statement x. By applying a semantics to the graph we find out that A_1 is IN and A_2 is OUT. What should be the correct label for the statement x? Of course, different strategies are possible – for example to consider the statement IN if there is at least one IN argument for it – and this final block in the argumentation pipeline – the *Statements Labeller* – is responsible for the application of the chosen strategy.

Pipeline. It is worth highlighting that not all the steps in the pipeline are always required: yet, considering only sub-portions of the pipeline, a broader range of argumentation scenarios could be modelled. For example, by considering the *Graph Transformer* and the *Argument Labeller* blocks only, we map the abstract argumentation [Dung, 1995a] where arguments are considered as atomic entities without an internal structure. Here, a transformation for the graph or the application of a semantics to it can be provided. As a further example, we can just consider a pipeline containing only the last block—the *Statements Labeller*. In this case a mechanism for evaluating statements without the need for arguments can be provided—i.e. the pipeline would be acting as a *defeasible logic* reasoner [Nute, 2001].

Summing up, the pipeline represents a generic structure that can be customised in order to fit the required argumentation model and scenario.

5.1.2 Requirements

We argue that the main goal of a general-purpose argumentation tool is to enable an easy definition of the above described custom pipelines, thus working just as an orchestrator for the entire argumentation process. Under this perspective, we could interpret every step in the pipeline as a different module in the tool, each one with different responsibilities, yet always referring to a single step in the argumentation process. It follows that the main requirement of an argumentation technology should be:

(Req1) enabling an easy definition of new modules;

(Req2) providing a way to define user-specific pipeline using modules;

(Req3) providing a mechanism responsible for the correct execution of the entire process;

(Req4) providing a common medium on which modules can exchange their input/output data;

(Req5) ensuring the traceability and accountability of the entire process.

The next sections discussed how these requirements have been implemented in Arg2P according to two main principles: *encapsulation* and *modularity*.

5.2 The Arg2P technology

Given the final aim of our research – that is, to provide a comprehensive and innovative tool for spreading intelligence and argumentation capabilities in nowadays challenging AI context such as IoIT [Arsénio et al., 2014] – we devote our attention to computational logic as the foundation for our work. With respect to available LP languages, Prolog represents the most successful one as a well-defined language coming with several implementations. For this reason, logic-based technologies are typically either built on top or as extensions of the Prolog language. However, existing solutions mostly work as monolithic entities tailored upon specific inference procedures, unification mechanisms, or knowledge representation techniques. Instead, considered the ultimate goal of our research, we require a technology supporting and enabling the exploitation of all the manifold contributions from LP. For this reason we choose to build Arg-tuProlog on the top of tuProlog—and in particular on 2P-Kt, a reboot of the tuProlog [Denti et al., 2005] project offering a general, extensible, and interoperable ecosystem for LP and symbolic AI [Ciatto et al., 2021].

Leveraging on 2P-Kt – and thanks to its architecture (detailed in Section 5.1) – , the Arg-tuProlog engine offers a neutral ground upon which different technologies can be integrated for building transparent and explainable systems. The rationale behind this choice is to enable the incremental addition of novel functionalities to the engine, possibly targeting other argumentation/conversation strategies, while supporting as many programming platforms as possible. In the following the main key technology features, reflecting the desired requirements, are summarised.

Interoperability & portability.. The interoperability requirement is guaranteed by the choice of Prolog – tuProlog in particular – as the main technological foundation of the solver. Indeed, the exploitation of the Prolog paradigm ensures the maintenance of the maximum degree of standardisation thanks to the ISO standard.[2] Besides, the Kotlin-based engine of 2P-Kt – devoted to heavy-interconnected and pervasive contexts – enables the system to run in more disparate environments. Accordingly, the Arg-tuProlog framework natively supports

[2] https://www.iso.org/standard/21413.html

interoperability with JVM, JavaScript and Android platforms.

Modularity & customisation.. The entire Arg-tuProlog framework is a collection of tuProlog compatible libraries—thus enforcing system modularity, as discussed in Section 5.1. 2P-Kt features are exploited to allow external libraries to be included during the evaluation process. The software organisation through distinct libraries ensures on the one hand the modularity and separation of concerns – fundamental pillars of a solid, easily maintainable and extensible technology –; on the other hand, it offers the simplicity of customisation in presence of domain-specific requirements. As a consequence, the Arg-tuProlog engine makes it possible to exploit diverse programming paradigms and technologies for any system component. In the current implementation, Arg-tuProlog acts as a meta-interpreter that accepts theories in a well-defined argumentation language, then, once translated in Prolog, provides the solutions. Consequently, it would be quite easy – for example – to add a distinct module exploiting a SAT-solver to compute the labelling over an entire argumentation graph.

Light-weightness & generality.. In contrast to the current trend of building highly-specialised systems, the Arg-tuProlog framework places itself as a highly-general tool to be injected in most disparate environments for the most different application purposes. As shown in [Calegari et al., 2021a], the MAS community is gazing for interoperable and general-purpose logic-based technologies: there, the Arg-tuProlog engine, along with 2P-Kt, provides a technological substrate supporting agents' reasoning and conversation via manifold strategies. Moreover, thanks to the deep customisability of tuProlog, the Arg-tuProlog engine can be potentially exploited also within constrained systems with strict requirements in terms of available computational resources and limited memory footprint—as in the case of the typical IoIT scenarios [Arsénio et al., 2014].

The engine is available as a standalone application (Arg-tuProlog Java IDE), and as a 2P-Kt library (Kotlin library) [Ciatto et al., 2021, Ciatto et al., 2020]. All the release are available on the Arg-tuProlog GitHub repository.[3]

[3] `https://github.com/tuProlog/arg2p-kt/releases`

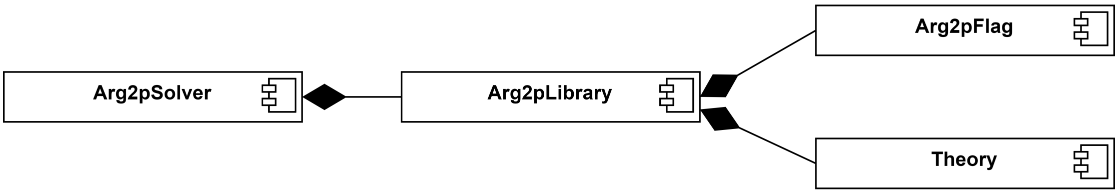


Figure 5.2: The Arg2P architecture.

5.2.1 Arg2P as a 2P-Kt extension

Figure 5.2 shows the general architecture behind the Arg2P[4] framework and its main components: `Arg2pSolver`, `Arg2pLibrary`, `Arg2pFlag`, and `Theory`. The core concept underpinning the Arg2P technology is the one of `Arg2pLibrary` – we could say that everything in the framework is a library – i.e., a collection of functions and procedures (`Theory` in Figure 5.2), and of a set of flags (`Arg2pFlag`). Then, a `Arg2pSolver` is composed by one or more `Arg2pLibrary`. We can then define the Arg2P framework as a composition of different libraries.

The ratio behind this choice is to meet the flexibility feature (in terms of ease of extension) in order to guarantee the possibility to encapsulate the many existing argumentation theories. Designing the framework as a composition of different libraries – where a library could implement either a single step of the pipeline or many of them at the same time or the communication medium or also the pipeline orchestrator – we enforce the flexibility feature decoupling each component from the others.

In the following we will discuss how the Arg2P technology meets all the requirements discussed in Section 5.1. Figure 5.3 shows how the general architecture presented in Section 5.1 is reified in the 2P-Kt technology.

The *library* entity is transposed to the `ArgLibrary` interface (Listing 5.1) composed of three elements: an `alias`, the library identifier; the `baseContent`, the set of functions and primitives that compose the library; and the `baseFlags`, a set of directive that can be used to customise the behavior of the library.

Listing 5.1: Arg2P library.

```
interface ArgLibrary {
```

[4] The sources are available at `https://github.com/tuProlog/arg2p-kt`

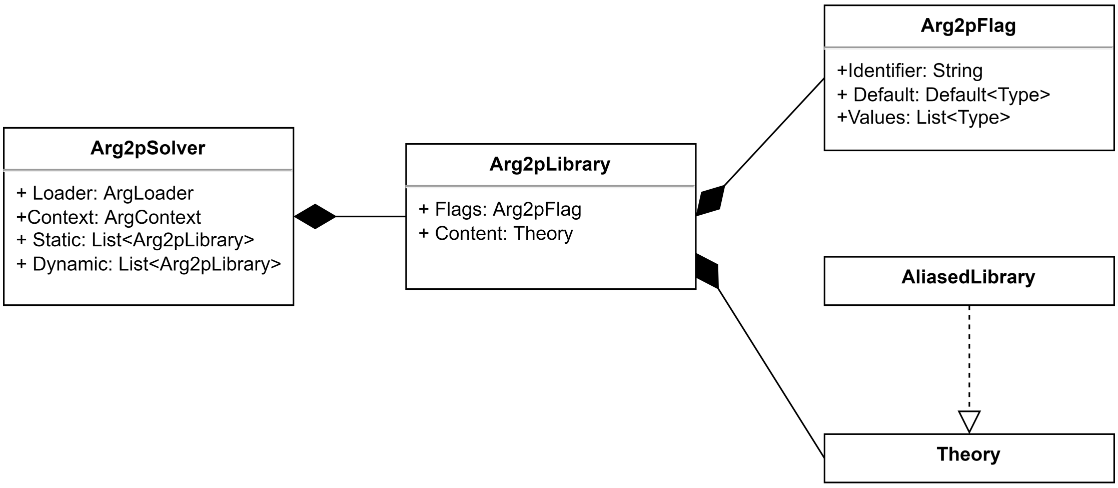


Figure 5.3: Arg2P in tuProlog.

```
    val alias: String
    val baseContent: AliasedLibrary
    val baseFlags: Iterable<ArgsFlag<*, *>>
}
```

The `baseContent` is declared as `AliasedLibrary`. This is one of the abstractions available in the 2P-Kt environment to easily declare a plug-in extension for the base Prolog solver. Every `AliasedLibrary` contains a list of operators, predicates and/or functions—encoded in Java, Kotlin or Prolog. Adding a library to a 2P-Kt solver means adding the content as a static knowledge base—i.e., all the predicates and functions added to the solver through libraries will be callable during the solving procedure as declared in the input theory. The use of the `AliasedLibrary` primitive as the base of an `ArgLibrary` allows the tool to meet the requirement (Req1).

The *flag* entity is transposed to the `ArgsFlag` interface (Listing 5.2) representing a parameter that can be set in order to customise the behavior of a library.

Listing 5.2: Arg2P flag.

```
interface ArgsFlag<T, G> {
    fun predicate(): String
    fun default(): T
    fun values(): G
}
```

Their use recalls the one of C preprocessor directives: according to the chosen directives, different sections of the original source code will be considered during the final evaluation. Accordingly, a developer of a library can declare a set of flags that will be available to end-users in order to dynamically customise the computation. The structure of every `ArgFlag` is pretty simple: it contains a `predicate`, the identifier of the flag that will be used as the functor of the flag predicate in the logical theory, a default value and the list of allowed values. The use of flags to customise the computation, combined with ease of extension (intrinsic to the use of libraries), allows the tool to meet the second requirement (Req2).

Libraries are used in the final Arg2P solver in different ways, the distinction is made clear in the `Arg2pSolver` interface (Listing 5.3).

Listing 5.3: Arg2P solver.

```
interface Arg2pSolver {
    val loader: ArgLoader
    val context: ArgContext
    fun staticLibraries(): Iterable<ArgLibrary>
    fun dynamicLibraries(): Iterable<ArgLibrary>
}
```

Two categories of libraries can be identified – static and dynamic – as well as two additional elements, the `ArgLoader` and the `ArgContext`.

The distinction between static and dynamic libraries has been introduced to cope with private routines and data, enabling – via the dynamic option – the creation of a new solver with the specified knowledge base loaded. Conversely, Prolog – so, 2P-Kt – does not offer a native mechanism for encapsulating private routines and data—thus making it difficult to avoid naming collision in an open development scenario. Accordingly, *static* components are the ones that should always be loaded in the solver – as in standard 2P-Kt – and their primitives should be available during all the resolution procedures. For instance, the user operational interface could be included in the solver as a static library. Conversely, *dynamic* components have their own namespace and context, and thus their execution is separated from other dynamic libraries, thus enforcing a form of strong encapsulation and information hiding. The framework mechanism for dealing with this form of dynamic loading of a library is represented by the `ArgLoader` component. Details of the available implementation of the loader are discussed in Subsection 5.2.1. The `ArgLoader` component allows the framework to meet the (`Req3`) requirement by providing a mechanism to guarantee the soundness of the logical solver built-in in the 2P-Kt environment.

Finally, the `ArgContext` component provides an additional method of communication than the "function" calls, so that the libraries can really be independent of each other. The `ArgContext` represent a common medium enabling the communication between the libraries, requirement (`Req4`). Moreover, tracking the data on this medium would improve the transparency of the entire process, meeting requirement (`Req5`). Implementation details on this shared context are discussed in Subsection 5.2.1.

The Context Library

The general idea behind the context library is to provide a *shared medium* where framework components can save their data thus providing a completely independent interaction model. For instance, a library implementing the *Argument Labeller* module would not need to call directly a *Graph Builder* component to obtain a graph (or viceversa), but the data should already be available in its context. The current Arg2P framework implements the mechanisms through a git-based branching model.

The context is implemented as a 2P-Kt solver with an empty knowledge base. It is possible to add, remove, and query the data inside the context through the following directives:

- `context_assert(Data)`, requesting the *Data* to be added to the context;

- `context_retract(Data)`, requesting all the data matching *Data* to be deleted from the context. Being the context based on a Prolog solver *unification* and *variables* can be exploited to match more than one entry in the context;

- `context_check(Data)`, checking for all the entries in the context matching

Data. In case of more than one match the results are given via backtracking.

These three directives offer the base functionalities required to a standard cache. Moreover, some directives have been added to track the evolution of the context over time thus enabling to return to a given point in the context history. This feature enforces the transparency of the argumentation process for end-users (`Req5`)—possibly to easily check the behaviour of the single pipeline components. W.r.t. the latter, indeed, many argumentation algorithms require a full exploration of the search space to find the right solutions. However, the *backtracking* capabilities offered by the base Prolog solver are not enough to deal with the data in the knowledge base—i.e., asserted data survive backtracking. A branching context is one of the possible solutions to the problem: by branching the context it is possible to explore the search branch and in case of necessity seamlessly return to its original state.

The *Context* library makes available five directives to this purpose:

- `context_active(Context)`, returns the identifier of the active context;
- `context_checkout(Context)`, requires the activation of the context with the `Context` identifier;
- `context_branch(Context, NewContext)`, creates a new context identified by *NewContext* starting from the one identified by *Context*. This is easily done in creating a new 2P-Kt solver with the same knowledge base of the *Context* associated solver;
- `context_check(Context, Data)`, checking for all the entries in the context identified by *Context* matching *Data*;
- `context_reset`, reset the context history and create a new empty context.

The Loader Library

The last element composing the Arg2P framework is the *ArgLoader* library, responsible for the loading and execution of dynamic components. The engine offers a base implementation exploiting 2P-Kt solvers. When the execution of a primitive in a dynamic library is requested, then:

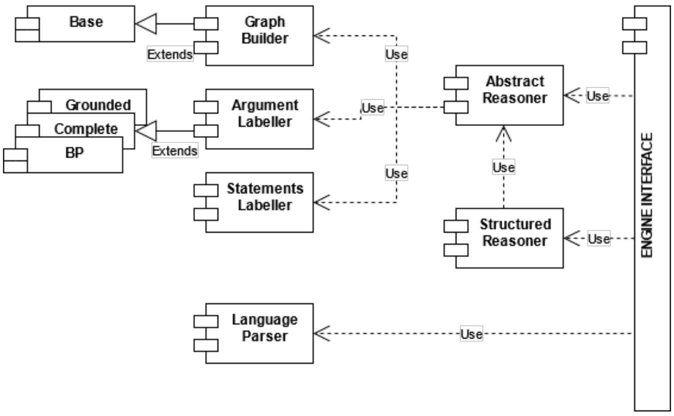


Figure 5.4: The ASPIC$^+$ libraries for Arg2P.

1. a new 2P-Kt solver is created, containing only the defined *static* libraries and the target *dynamic* library;

2. the primitive is executed by the new solver: by default, the same context is used inside the new solver;

3. when the execution ends, the newly created solver is discarded and the original environment restored; the only side effect concerns the execution context: if the "dynamic call" has produced changes in the context – e.g. change in the data, branching of the context –, they will be observable in the caller environment, too.

A primitive in a dynamic library can be called using the `::` operator. For example, `targetlib::targetfunction` executes the `targetfunction` from the `targetlib` library using the loader mechanism. There could be cases in which the caller does not intend the call to affect its state via changes in its context. In that case, the *ArgLoader* library offers another operator – `:::` – that automatises the branching of the context when the dynamic call is made. As a result, it is always possible for the caller to restore its original context and proceed as if the call never happened.

5.3 An Arg2P Instance: ASPIC and more

In this section, a description of the tools currently implemented in the Arg2P framework is provided.Figure 5.4 shows an Arg2P instantiation for an ASPIC$^+$-

like framework [Modgil and Prakken, 2014a].

The engine is designed in a fully-modular way, as depicted in Figure 5.4. Every function inside the framework – e.g., graph building, argument labelling – is sealed within the corresponding module. From a software engineering perspective, modularity highly improves the upgradability and flexibility – in terms of feature addition or maintenance – of the whole system.

One of the distinguishing aspects of the engine is represented by its design, which aims at providing two distinct ways of use:

1. the *graph-based* mode providing as output the entire argumentation graph according to the specified semantics—i.e., the labellings of the entire set of facts and rules given as input;

2. the *query-based* mode providing as output the evaluation of a single query given as input and according to a given semantics—i.e., enabling defeasible reasoning on arguments starting from certain premises.

While the former mode can be considered as the traditional approach of argumentation tools, the latter makes the engine framework a fit choice for the AI pervasive scenarios.

In the following, we describe each component in detail.

5.3.1 Engine interface

The *Engine Interface* module is the main pillar of the framework: it manages all the interaction with external entities (both software agents and humans) by providing a set of API to perform operations. Accordingly, the module hides all the complexity of the framework, but, at the same time, ensures all the custom semantic features to be set. The module exposes only the two usage predicates – *answerQuery* and *buildLabelSets* – transparently coordinating the core modules.

The predicate `buildLabelSets` builds the argument labelling and the statement labelling according to the theory. It first builds the argumentation graph, then evaluates arguments and statements. In particular, `IN`, `OUT`, and `UND` sets are created by classifying both arguments and statements accordingly to the specified labelling semantics—respectively, `IN`Arg, `OUT`Arg, and `UND`Arg for arguments and

INFact, OUTFact, and UNDFact for statements. The IN set includes admissible arguments; the OUT set includes the rejected ones; the UND includes those for which it was not possible to affirm the admissibility or not—possibly due to lack of or contrasting information. There are three variants for this predicate:

- *buildLabelSets/2* – `buildLabelSets([-INArg,-OUTArg, -UNDArg], [-INFact,-OUTFact,-UNDFact])` – returns both the sets of statements and arguments;
- *buildLabelSets/3* – `buildLabelSets(-IN, -OUT, -UND)` – where the output sets refer only to the statements clustering;
- *buildLabelSets/0*, which does not provide any output, but performs the argumentation graph construction and evaluation in background, and prints argument and statement sets as plain text.

The predicate `answerQuery(+Goal, -Yes, -No, -Und)` requests the evaluation of the given *Goal*. The output corresponds to the set of facts matching the goal, distributed in the three sets IN, OUT, and UND always accordingly to the labelling semantics.

Finally, the engine interface provides flags for customising the resolution process:

- `argumentLabellingMode(MODE)` sets the desired labelling semantics by means of on-purpose flags: namely, `grounded`, `complete`, `bp_grounded_partial`, and `bp_grounded_complete` (details in [Baroni et al., 2011b, Calegari and Sartor, 2020a]); future works will also involve further semantics implementations.
- `orderingPrinciple(MODE)` sets the strategy for the preference propagation over arguments: the admissible values are `last` and `weakest`;
- `orderingComparator(MODE)` sets the superiority relation to exploit for argument ranking: implemented strategies are `democrat` and `elitist`, both presented in [Modgil and Prakken, 2014a], and `normal` from [Dung and Thang, 2018];
- `queryMode`, if present, enables the structured evaluation of *answerQuery/4*, i.e., exploiting the pseudo-DeLP algorithm (the flag can be enabled only for grounded semantic);

- `autoTransposition`,when selected, enables the automatic computation of the base theory closure under transposition [Caminada and Amgoud, 2007c] before starting the evaluation process;

- `unrestrictedRebut` disables the restriction on rebut attack relation: by default, rebut attack relation is restricted—i.e. arguments having a strict rule as a top rule cannot be rebutted [Caminada and Amgoud, 2007c]; so, if the user selects this flag, the restriction does not hold anymore.

Two more flags are introduced in the interface, preparing the engine for future extensions and customisations:

- `graphBuildMode(MODE)` sets the argumentation modality of the framework (for instance, base, argumentation over preferences, meta-argumentation); currently, according to the above-discussed model, only `base` can be selected, but future works will be devoted to new modalities implementation;

- analogously for the `statementLabellingMode(MODE)` with respect to the statements labelling step; currently the only implemented modality is `base`—i.e. arguments' labels transfer to their claims;

5.3.2 Language Parser

The *Language parser* is a support modules, bearing transversal responsibilities w.r.t. the entire framework: its role is to convert the rules from the argumentation language of the framework to an internal representation more efficiently exploitable by the engine. Moreover, the exploitation of a distinct internal representation enforces the framework flexibility, decoupling the user and the technical language thus encouraging a continuous evolution of both the language and the computational mechanisms.

The engine adopts an ASPIC$^+$-like syntax – introduced in [Modgil and Prakken, 2014a] – and exposes all the main features of this framework.

In the following the language predicates are discussed.

Rules & premises

Strict rules can be expressed exploiting the −> operator, with the notation:

$$\mathit{label} : \mathit{premise_1}, \ldots, \mathit{premise_n} \; {-}{>} \; \mathit{conclusion}.$$

where *label* is the identifier (unique name) of a rule and $\mathit{premise_1}, \ldots, \mathit{premise_n}$ are the premises that entails the *conclusion*. Accordingly, axioms take the form:

$$\begin{aligned} &\mathit{label} \quad : \; {-}{>} \; \mathit{conclusion}. \\ (&\mathit{label} \quad : [\,] \; {-}{>} \; \mathit{conclusion}.) \quad \text{(this form is also permitted)} \end{aligned}$$

Defeasible rules are expressed by the => operator:

$$\mathit{label} : \mathit{premise_1}, \ldots, \mathit{premise_n} \; {=}{>} \; \mathit{conclusion}.$$

Accordingly, defeasible premises take the form:

$$\mathit{label} \quad : \; {=}{>} \; \mathit{conclusion}.$$

Note that in the case of defeasible rules the rule

$$\mathit{label} \quad : [\,] \; {=}{>} \; \mathit{conclusion}.$$

even if admissible, is not equal to a premise, i.e., can affect differently the ordering relation over arguments and can be undercut by other rules.

Premises and conclusions can take any form containing compound terms, variables, and strong negations.

Attack relations

The binary attack relation between arguments – which underpins rebutting and undermining (rebutting on premises) attacks – can be reached via term negation. Two types of negations are available:

- $-\mathit{term}$, to indicate a strong negation (contrary), that captures a notion of negation as definite falsity—i.e., the strong negation of a formula entails its intuitionistic negation;

- $\sim (term)$ to indicate weak negation (negation as failure).

Weak negation is allowed only inside rules premises, representing an exception to the universal rule. Moreover, strong and weak negations cannot be nested.

Undercut attacks can be expressed exploiting the notation:

$$\mathit{label_2} : \mathit{premise_1}, \ldots, \mathit{premise_n} => \mathit{undercut}(\mathit{label_1})$$

where $label_1$ is the identifier of a defeasible rule in the theory. So, for instance, let r_1 be a rule stating that things that look of a certain colour (let it be red) are usually of that colour. And let r_2 be a rule stating that objects illuminated by a coloured light (let it be red) look of that colour even if they are of a different colour. The corresponding knowledge and the undercutting relation between the two rules can be expressed as:

$$\begin{aligned} r_1 :&\quad \mathit{look}(\mathit{Object}, \mathit{Colour}) => \mathit{colour}(\mathit{Object}, \mathit{Colour}). \\ r_2 :&\quad \mathit{illuminated}(\mathit{Object}, \mathit{Colour}) => \mathit{undercut}(r_1). \end{aligned}$$

The reason is that there is a counter-argument that can undercut the original argument by attacking the connection between the claim and the reason.

Is it possible to define custom conflicts through the predicate $\mathit{conflict}/2$:

$$\mathit{conflict}([Term1], [Term2]).$$

Using this syntax will be possible to modify the conflict relation used to in ASPIC$^+$ to define attacks between arguments. For example, using this syntax, strong negation can be defined as:

$$\begin{aligned} &\mathit{conflict}([-Term], [Term]). \\ &\mathit{conflict}([Term], [-Term]). \end{aligned}$$

Superiority relation

It is possible to denote preferences by using the following notation:

$$sup(ruleName_1, ruleName_2)$$

This proposition states that the rule (non-axiom premise) with identifier equal to $ruleName_1$ is superior to the one with identifier $ruleName_2$.

5.3.3 Core modules

In addition to the *Engine Interface* and *LanguageParser* modules, other modules depicted in Figure 5.4 bear different responsibilities. The core of the engine contains the *Abstract Reasoner* and the *Structured Reasoner* modules. As mentioned above, the engine exposes two distinct operation modes – i.e. *graph* and *query-based* mode. These two modules are responsible for orchestrating these functionalities: the former is held by the *Abstract Reasoner* module while the latter by the *Structured Reasoner*.

These two modules are not designed as monolithic entities but can leverage on other sub-modules, categorisable in a third group—namely, the algorithmic group. Each module belonging to this group is focused on a single algorithmic responsibility. In particular, existing algorithmic modules are:

- *Graph Builder* builds the argumentation graph starting from a rule-base encoded with the engine internal representation

- *Grounded Labeller*, in charge of computing grounded labelling of the argumentation graph, according to Dung's notions of grounded semantics

- *Complete Labeller*, in charge of computing complete labelling of the argumentation graph, according to Dung's notions of complete semantics

- *BP Labeller* builds the second stage burden of persuasion labelling starting from the grounded labelling

- *Statement Labeller* carries out the statements labelling according to labelling of the arguments.

It follows that, by simply adding or changing a module, the user can completely change the behaviour of the engine and adapt it to contingent and domain-specific requirements (such as supporting a new semantic) offering an alternative customised implementation for a building block, or supporting a custom argumentation language.

Let us assume as input a pipeline made of the libraries depicted in Figure 5.4: then, the steps required to create an Arg2P instance are discussed in the following. First of all, an *ArgLibrary* for the *Abstract Reasoner* module should be created—representing the pipeline orchestrator. Listing 5.4 shows a possible implementation in Kotlin.

Listing 5.4: An AbstractMode library.

```
sealed class AbstractMode : ArgLibrary {

    override val alias = "prolog.argumentation.abstract"

    override val baseContent: AliasedLibrary
        get() = Library.aliased(
            alias = this.alias,
            theory = prologTheory
        )

    override val baseFlags: Iterable<ArgsFlag<*, *>>
        get() = listOf(GraphExtension, ArgumentLabellingMode
           ,
            StatementLabellingMode, GraphBuildMode)
}
```

The implementation contains the library identifier – `prolog.argumentation.abstract` –, the list of the flags that the library accepts – e.g. `ArgumentLabellingMode` defining the semantics to be applied to the graph, or `GraphExtension` declaring the available graph transformations –, and then the implementation is provided through the `AliasedLibrary` primitive. For the sake of simplicity, we assume that the Prolog content is given as raw text through the `argTheory` variable containing just a list of clauses using the flags to determine the library to invoke in order to complete the argumentation process.

The creation of the solver requires the list of the static and dynamic libraries to be exploited in the resolution procedure—Listing 5.5.

Listing 5.5: Arg2P solver.

```
fun arg2p(): Arg2pSolver = Arg2pSolver.of(
    listOf(EngineInterface),
    listOf(
        AbstractMode, RuleParser, ArgumentationGraphBuilder,
        ...
    )
)
```

In the case at hand, there is only one static library, the *EngineInterface*. It exposes the main predicate to interact with the framework – *buildLabelSets*/3 – that requires for the execution of the entire pipeline: taking the rules in the knowledge base as input and returning the statements labelling as output—IN, OUT or UND sets.

The second list contains all the dynamic libraries required by the evaluation. In the case at hand, we have the *AbstractMode* library – acting as an orchestrator of the entire pipeline –, and the *RuleParser* and *ArgumentationGraphBuilder*, respectively responsible for the first and the second steps in the general pipeline.

Once an instance of the Arg2P framework is defined, it is possible to create a 2P-Kt solver to carry on the computation:

Listing 5.6: 2P-Kt solver.

```
val solver = ClassicSolverFactory.
   mutableSolverWithDefaultBuiltins(
    otherLibraries = arg2p().to2pLibraries()
        .plus(FlagsBuilder().create()),
    staticKb = argTheory
)
```

The *arg2p* instance is transformed in a list of 2P-Kt libraries through the `to2pLibraries` method. Also, a library containing the user defined flags is added to the list[5]. The

[5] `FlagsBuilder` is a utility class that creates a library of flags starting using the data with

theory to be used in the argumentation process is provided to the system through the `argTheory` variable.

The final evaluation can so be performed (Listing 5.7):

Listing 5.7: Query evaluation.

```
val solution =  arg2pScope {
    solver.solve("buildLabelSets"("StatIn", "StatOut", "
       StatUnd")
}.first()
```

The *EngineInterface* is the only statically-defined library, thus making the *buildLabelSets*/3 the only predicate in the global namespace. By requiring its evaluation, we obtain a solution containing a Prolog substitution for the three variables "StatIn", "StatOut", "StatUnd" according to the input theory. Thanks to the *Context* library, it is then possible to navigate all the steps in the resolution process through the directive `context_checkout/1`.

5.3.4 Deontic rules and conflicts

Our deontic extension focuses on basic concepts of deontic reasoning—namely, obligations, prohibitions, and permissions. Obligations are at the core of our deontic system, and prohibitions are viewed as a by-product of obligations. As stated in [Riveret et al., 2019], in this context we can say that *something is prohibited* is equivalently expressed by stating that *its opposite is obligatory*. Also, permissions can be reloaded in terms of obligations: permission to do something expresses that the opposite is not obligatory. Accordingly, and for the sake of simplicity, the attention is restricted to a propositional language which is supplemented with a single deontic operator O which indicates an obligation. Hence, we assume a language whose literal statements are enhanced with the following definition of deontic literal statements:

Definition 48 (deontic literal statement)**.** *A* ***deontic literal statement*** *is a statement of the form $O\gamma$ or $\neg O\gamma$ such that γ is a plain literal statement. Prohibitions and permissions are captured by assuming that a prohibition $F\gamma$ is equiva-*

which has been fed. In the example, the library is created only with flags' default values.

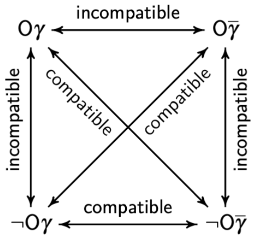


Figure 5.5: Deontic square of compatibility relation

lently expressed by the obligation $O\gamma$, and a permission $P\gamma$ is syntactically equivalent to $\neg O\gamma$.

Deontic operators extend the semantics of the rule by enabling the definition of the so-called *normative rules*, i.e., containing either normative concepts or deontic operators. The definition of these rules affects and enlarge the conflict relation of the argumentation framework. According to the deontic semantics, deontic conflicts have to be considered, too, namely conflicts of the form $(\gamma, \bar{\gamma})$ or $(O\gamma, O\bar{\gamma})$ or $(\neg O\gamma, O\gamma)$ or $(O\gamma, \neg O\gamma)$ as depicted in the deontic square in Figure 5.5. Detailed formal accounts of the adopted deontic extensions are discussed in [Riveret et al., 2019]—our model fully adheres to that semantics.

In the language of the framework we have:

- $p(term)$ to indicate permission;
- $o(term)$ to indicate obligation.

Note that we introduced the p functor to lighten the notation, but it perfectly fits the model discussed in Subsection 5.3.4 since a permission $P\gamma$ is syntactically equivalent to $\neg O\gamma$.

Prohibitions can be defined exploiting strong negation: $-p(term)$. Consequently, a lack of prohibition can be defined as exploiting two strong negations: $-o(-term)$. This is the only exception to negations nesting prohibition (in general not allowed by the argumentation language).

5.3.5 Burden of Persuasion

The burden of persuasion on a proposition can be expressed as follows:

$$bp(term_1, \ldots, term_n)$$

The structure of terms reflects the one seen for standard rules: compound terms, variables and strong negations are therefore allowed.

The algorithm described in Section 4.3 has been tested and implemented in the Arg2P framework[6] [Pisano et al., 2020, Pisano et al., 2021]. Please note that the equivalence of the optimised procedure with the formal model presented in Chapter 3 has for now only been conjectured, thus remaining still unproved. Figure 5.6 displays the tool in action as it evaluates an example involving a conditional BoP constraint.

The entire process is based on grounded semantics and reachability checking—both polynomial complexity [Kröll et al., 2017b]. The algorithm requires $m + 1$ evaluation stages to end – where m is the number of connected burdened arguments –, then the final complexity is polynomial.

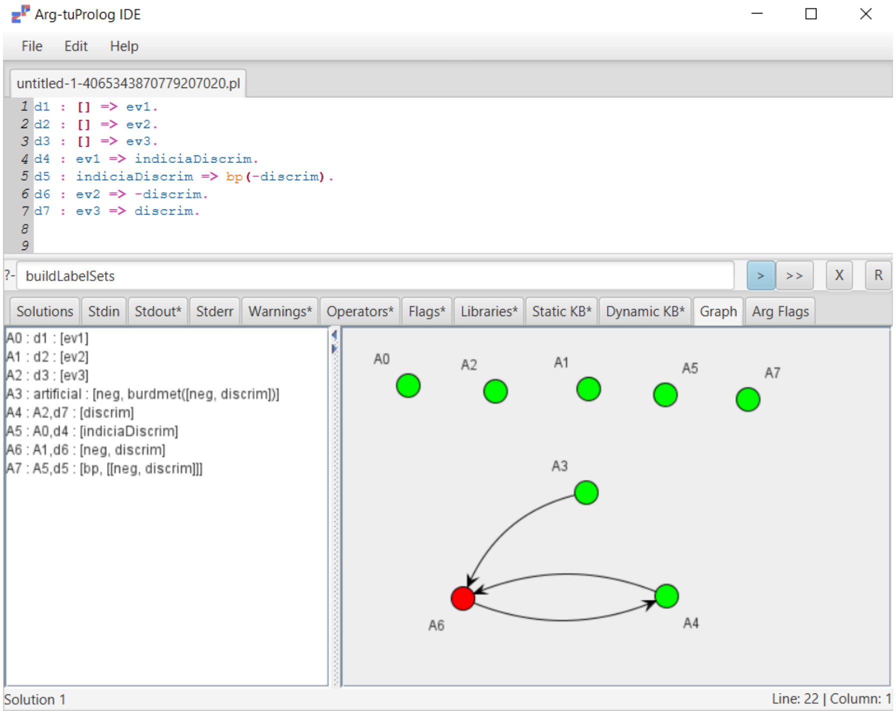
5.3.6 The parallel library

The model in Subsection 4.2.1 has been implemented as a library – the *Parallel* library – for the Arg2P framework. The goal of the implementation is twofold: *i)* provide a mechanism for the concurrent evaluation of a claim by a single Arg2P instance – actors in execution on a single machine can achieve real parallelisation thanks to multicore hardware architectures – *ii)* enable cooperative argumentation by allowing different Arg2P instances to create a single actor system, thus sharing their knowledge base or their hardware resources.

Among the available technologies for the implementation we selected Akka[7] [Cossentino et al., 2018]. Akka is an open source middleware for programming concurrent and distributed actor systems based on the original Actor model by Hewitt [Hewitt et al., 1973]. Build upon the JVM platform, the framework offers

[6] `http://arg2p.apice.unibo.it/`

[7] `https://akka.io/`



```
d1 : [] => ev1.
d2 : [] => ev2.
d3 : [] => ev3.
d4 : ev1 => indiciaDiscrim.
d5 : indiciaDiscrim => bp(-discrim).
d6 : ev2 => -discrim.
d7 : ev3 => discrim.
```

```
buildLabelSets
```

```
A0 : d1 : [ev1]
A1 : d2 : [ev2]
A2 : d3 : [ev3]
A3 : artificial : [neg, burdmet([neg, discrim])]
A4 : A2,d7 : [discrim]
A5 : A0,d4 : [indiciaDiscrim]
A6 : A1,d6 : [neg, discrim]
A7 : A5,d5 : [bp, [[neg, discrim]]]
```

Figure 5.6: Arg2P Meta-BoP evaluation

an easy way of deploying network distributed systems observant of the original actor principles—e.g. reactivity, asynchronous communications, absence of states of shared memory between actors. All these features made the Akka framework one of the reference technologies in the distributed landscape, with users ranging from the research to commercial fields.

The final implementation makes use of the *Akka Clustering* features to enable the collaboration of different Arg2P instances. In particular, we rely on *Cluster Singletons*[8] to handle the `Master` actor lifecycle, and *Cluster Sharding*[9] for `Worker` nodes. The *Parallel* library makes available five directives:

- `join(Port)`, requesting the creation of an actor system on the local machine exposed on port `Port`;

- `join(Port, Address)`, to join an actor system on the machine at the given `Address`, exposed on port `Port`;;

[8] `https://doc.akka.io/docs/akka/current/typed/cluster-singleton.html`

[9] `https://doc.akka.io/docs/akka/current/typed/cluster-sharding.html`

- `load`, requesting the distribution of the rules contained in the knowledge base of the local instance between all the members of the actor systems (Figure 4.2);

- `reset`, requesting the deletion of the data previously distributed in the actor system via the `load` directive;

- `solve(Goal, In, Out, Und)`, requesting the evaluation of the `Goal` claim to the actor system according to the procedure in Figure 4.2. We have the set of facts matching the goal distributed in the three sets IN, OUT, and UND as a result.

All the application scenarios can be modelled by using the directives above. We achieve parallel evaluation of a claim on a single Arg2P instance in three steps: *(i)* creating a local actor system (`join(Port)`), *(ii)* distributing the theory between local actors (`load`), *(iii)* requiring the evaluation of a statement through the `solve(Goal, In, Out, Und)` directive. At the same time we could have others Arg2P instances offering their hardware resources (`join(Port, Address)`) – if a node joins the actor systems the *Cluster Sharding* functionality will be responsible for the fair distribution of `Worker` actors between all the available nodes –, or also participating to the resolution if they share their own knowledge (`load`). Listing 5.8 shows an example of parallel resolution using a single node.

Listing 5.8: Arg2P parallel solver.

```
arg2pScope {
    ClassicSolverFactory.mutableSolverWithDefaultBuiltins(
        otherLibraries = Arg2pSolver.parallel().
           to2pLibraries(),
        staticKb = theory
    ).also {
        it.solve(
            "join"(2551) and
            "load" and
            "solve"("query", "In", "Out", "Und")
        )
    }
}
```

Chapter 6

Legal Reasoning

In this chapter, we delve into the application of our formal meta-argumentation model and the Arg2P technology within the legal domain. This exploration builds upon the theoretical frameworks established in previous chapters, aiming to demonstrate the real-world efficacy of these tools in addressing complex legal issues, thereby bridging the gap between advanced theoretical models and their tangible impact in the field of legal AI.

We begin by highlighting the application of our formal meta-model in the realm of legal studies, with a particular focus on legal interpretation. As demonstrated in the earlier section on the Burden of Persuasion extension, the formal advancements in our model have already shown a profound connection with the legal field. This section will serve as a prime example of how our theoretical developments are intricately linked with legal reasoning, further emphasising the relevance and applicability of our work in the legal AI landscape. We will demonstrate how the meta-model facilitates nuanced analysis and interpretation of legal texts and precedents. By applying the meta-model to interpretive challenges commonly faced in legal practice, such as dealing with conflicting precedents or ambiguous statutory language, we illustrate its ability to provide clarity and structure in complex interpretive tasks.

Following the focus on legal interpretation, we shift to illustrate how the principles of defeasible reasoning and formal argumentation, as encapsulated in our Arg2P technology, align seamlessly with the intricacies of legal reasoning. This chapter will showcase the adaptability of our approach in handling the dynamic and multifaceted nature of legal cases.

Central to this chapter are examples and case studies where our model and technology are applied to real-life legal scenarios. These examples will serve to highlight the practical utility of our tools in analysing and resolving legal disputes. We will demonstrate how our approach not only matches but also enhances the depth and clarity of traditional legal analysis, offering innovative solutions to longstanding legal challenges.

Overall, this chapter serves as a critical link between the theoretical advancements detailed in earlier sections, the practical applications in the field of legal reasoning, and their broader impact on the legal AI landscape.

6.1 Meta-Argumentation for Interpretative Reasoning

In general, Meta-argumentation plays a crucial role in the realm of legal reasoning, particularly when it comes to the process of justification. Legal reasoning is inherently dependent on the robust justification of contested legal statements, especially in controversial contexts. Firstly, the adequacy of justifying a legal claim goes beyond merely presenting an argument in its support. In legal reasoning, the premises of the argument themselves often require justification. This recursive need for justification continues until a point is reached where no additional justification is deemed necessary by the decision-maker. This highlights the need for a meta-argumentative approach where the validity and soundness of each layer of the argument are scrutinised and established. Secondly, the relevance of a criticism in legal argumentation is contingent upon its ability to present a claim that directly conflicts with the premise of the existing argument. Such criticisms are significant as their acceptance implies the rejection of the original argument. However, the perceived incompatibility between the criticism and the original argument can itself be a subject of debate. This leads to a situation where the compatibility or incompatibility of arguments becomes a central issue, requiring a meta-level analysis to resolve. Thirdly, the resolution of conflicts between arguments often involves preferences over these arguments. However, these preferences are not absolute and can be contested. The process of evaluating and establishing these preferences necessitates a meta-argumentation approach, where the reasons behind preferring one argument over another are thoroughly examined and justified. Intuitively, the requirements highlighted in the realm of legal reasoning and meta-argumentation are fully satisfied by the model we have presented. However, to elucidate this alignment more comprehensively, we will delve deeper into the specifics, using legal interpretation as our primary demonstration area. Our work is closely aligned with and expands upon prior efforts to develop argumentation-based models of interpretation, as seen in [Rotolo et al., 2015, da Costa Pereira et al., 2017, Walton et al., 2021].

Legal interpretation is a fundamental process in the practice of law, involving the methodical examination and elucidation of legal texts, such as statutes and judicial decisions. At the heart of legal interpretation are various interpretative canons, which are established methods or principles that guide legal practitioners in understanding and applying the law. While there are numerous canons, focusing on three primary ones — the *literal* canon, the *purposive* canon, and the *precedential* interpretation — can provide a clear insight into the core of legal interpretive practice.

The literal canon dictates that the interpretation of a legal text should be based primarily on the plain, ordinary meaning of the words used. It operates on the assumption that the language of the statute conveys the intent of the lawmakers. Legal practitioners and judges turn to this canon as a starting point, considering what the text explicitly states. The literal interpretation is especially favoured in situations where the language is clear and unambiguous, thus leaving little room for interpretation beyond the words themselves.

The purposive approach, also known as the teleological approach, goes beyond the mere wording of the text to consider the broader purpose and objectives behind the law. This canon is particularly useful in cases where a literal interpretation would lead to an absurd or unjust outcome, or where the intent of the law is not immediately clear from its language. It involves examining legislative history, the context surrounding the law's creation, and the overall goals the law seeks to achieve. This approach aligns legal interpretation with the broader objectives of justice and social welfare, ensuring that the application of law harmonises with its intended purpose.

Precedential interpretation relies heavily on past judicial decisions and interpretations to guide the understanding and application of legal texts. The doctrine of *stare decisis*, or the principle of adhering to precedent, plays a critical role here. Legal professionals look to previous rulings on similar issues as a guide for interpreting current cases. This approach ensures consistency and predictability in the law, as similar cases are treated in a similar manner. Precedential interpretation is fundamental in common law systems where past judicial decisions form a significant part of the legal framework.

Each canon offers a different lens through which the law can be understood and applied, often complementing each other to achieve a balanced and fair interpretation. Our discussion on legal interpretation will delve into how these canons are reflected in our formal argumentation model.

Generally speaking, modeling legal interpretation requires a system that can adeptly handle the complexities inherent in legal reasoning. Such a system needs to be capable of reasoning about various critical aspects that are central to interpreting statutory provisions and judicial rulings:

- the system must be capable of assessing the validity of rules that express interpretations of statutory provisions or judicial rulings. This involves not just understanding the rules themselves but also evaluating their legitimacy and applicability in different legal contexts. The model should be able to determine whether a particular rule is appropriate and valid for the interpretation of a given legal text;

- legal interpretation often involves conflicting views and interpretations. The system should be equipped to identify and reason about these conflicts. This includes recognising when different interpretations are at odds with each other and understanding the basis of these conflicts, whether they arise from differing interpretations of the law, conflicting precedents, or divergent applications of legal principles;

- in cases where there are multiple valid interpretations, the system must be able to reason about the preferences over these interpretations. This involves evaluating which interpretation should be given precedence based on legal principles, precedents, the context of the case, and the objectives of the law.

These requirements are satisfied by the model proposed in Chapter 3. In particular,

- any argument in which a rule is used will include a sub-argument supporting the validity of the rule—i.e., the legitimacy and applicability of the interpretation must be proved as well;

- any argument attacking another argument B will include a sub-argument for the contrariness between (aspects of) the two arguments—i.e., the source of conflict has to be both understood and agreed upon;

- any argument resisting an attack on the basis of its priority over the attacker, will include a subargument for the existence of the priority—i.e., the preference efficacy is determined by the context.

In the remainder of this section, we will present three distinct examples that effectively demonstrate the application of the meta-machinery in the context of legal interpretation. These examples are specifically chosen to illustrate the practical utility and effectiveness of our approach in navigating the complexities of legal reasoning.

Example 12 (Gender Identity)**.** *Let us consider a legal example concerning a case of gender identity. Let us consider the case of Sue. She wants to compete in the woman's chess championship but the organisers argue that this would be impossible because legally she has been assigned the male gender, as proven by her passport.*

However, Sue identifies herself as a woman and thinks that she should have the right to compete in the championship. To decide the case we should first decide on the existence of a conflict between the concepts of man and woman: are they in conflict – Sue's official gender automatically discards her claim of being a woman –, or can the two concepts coexist according to the principle of self-determination? To encode the case at hand, the argumentation model should allow conflicts to be formalised, i.e., a meta-argumentation model is required.

In introducing the case of Sue's eligibility to participate in a women's chess championship based on her gender identity, we venture slightly away from the direct application of traditional legal interpretative canons. However, this example remains highly relevant and insightful in showcasing the proficiency of our model in a complex legal issue intimately connected to interpretation.

This scenario challenges the model to navigate the intricate legal definitions and terms related to gender identity. While it may not involve the classical interpretation of textual nuances or legislative intent, it presents a real-world application of how legal texts and policies regarding gender are interpreted and applied. This exploration into statutory language and its practical implications reflects a key aspect of legal interpretation.

Furthermore, the case brings to the forefront the challenge of reconciling conflicting legal principles: the legal recognition as per official documents versus the principle of self-determination. This aspect of the case mirrors a common hurdle in legal interpretation, which involves resolving conflicts between different legal norms or principles. The model's capability to manage this conflict exemplifies its adeptness in interpreting and applying legal principles to specific situations, a critical skill in legal reasoning.

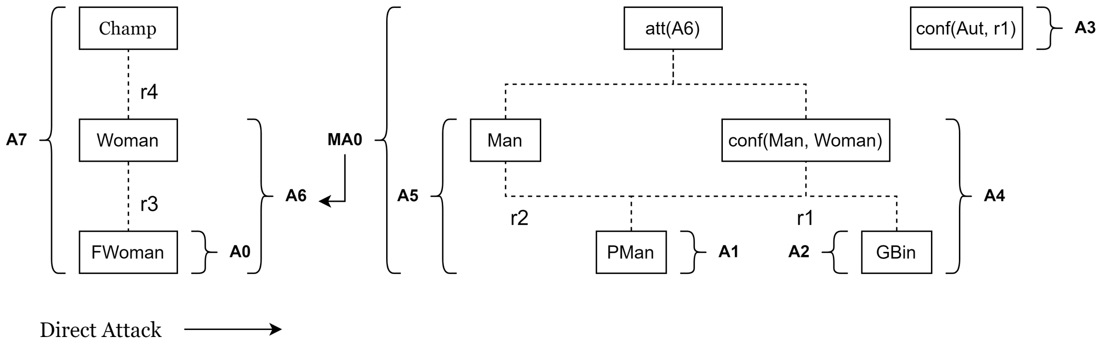
By examining this case, we aim to demonstrate how our model adeptly addresses issues that, while not strictly about interpreting legislative text, are deeply intertwined with the broader challenges of legal interpretation. It provides an insightful example of the model's capacity to dissect complex legal arguments, balance competing principles, and offer reasoned analysis, all integral elements in the practice of legal interpretation.

In this example we will make use of the parts of the model introduced in Section 3.2. We use the following abbreviations:

- Champ = Sue can compete in the women's chess championship
- FWoman = Sue identifies herself as a woman
- PMan = Sue's passport identifies her as a man
- Aut = Every person has the right to self-determine their gender

- GBin = Every person's gender is determined by their birth sex, either male or female

Example 13 (Gender Identity: Formalisation)**.** *Let us consider the theory where* $R_d = \{\ r1 : PMan, Gbin \Rightarrow conf(Man, Woman);\ r2 : PMan \Rightarrow Man;\ r3 : FWoman \Rightarrow Woman;\ r4 : Woman \Rightarrow Champ\ \}$ *with the following facts* $K_p = \{\ FWoman,\ PMan,\ GBin,\ conf(Aut, r1)\ \}$, $K_s = \emptyset$, $R_s = \emptyset$. *Accordingly to the above definitions, we can then build the following arguments:*



The attacks are $MA0 \rightsquigarrow A6$, $MA0 \rightsquigarrow A7$. *If we apply Dung's grounded semantics to the framework we obtain the extension* $\{A0, A1, A2, A3, A4, A5, MA0\}$*—i.e. the incompatibility between Sue's official gender and her perceived identity is confirmed (A4), preventing her to compete in the championship.*

Sue is not happy with the final decision and decides to appeal claiming that her right to self-determination has not been taken into due consideration. The case is evaluated again with the new information: $K_p = \{Aut\} \cup K_p$. *Two new arguments are obtained:*

A8 : Aut
MA1 : A8, A3 ⇒ *att(A4)*

The new attacks are $MA1 \rightsquigarrow A4$, $MA1 \rightsquigarrow MA0$. *Applying again Dung's grounded semantics to the framework we obtain* $\{A0, A1, A2, A3, A5, A6, A7, A8, MA1\}$*— i.e. the problem on Sue's identity is resolved discarding the conflict between her birth and perceived gender (A4), according to the principle of self-determination (A8). Consequently, she is free to compete in the championship. Indeed, the CAF was able to integrate the new knowledge and use it to revise the status of the*

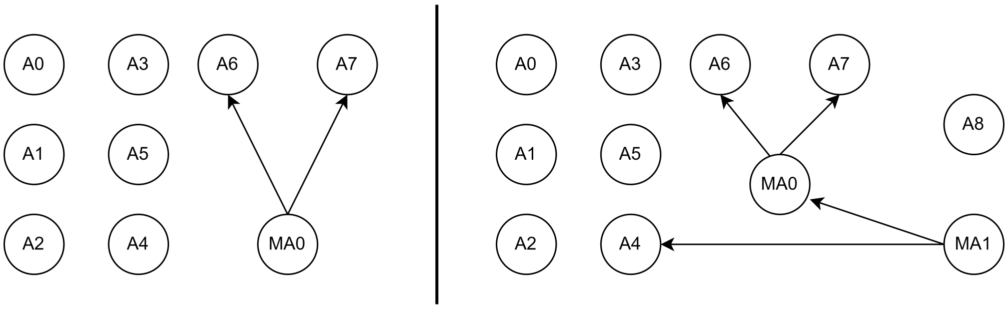


Figure 6.1: Conflict-based Argumentation frameworks from Example 13

propositional conflicts in the argumentation theory as expected. Both the original argumentation graph and the revised one are presented in Figure 6.1.

Let us now proceed with an interpretation example making use of the machinery from Section 3.4 and Section 3.1. Note that in this and in the following examples, we shall list rules and premises including variables $(X, Y, \ldots)$ as a abbreviations for the set of all their ground instances. In the arguments using such rules, we shall use the corresponding ground instances.

Example 14 (Interpretation - Indecent exposure)**.** *Let us assume that indecent exposures are prohibited (rule* $r0$*). This rule has been interpreted in such a way that breastfeeding counts as an indecent exposure (rule* $d1$*), and that this interpretation is considered to be valid, having been affirmed in a judicial decision (rule instance* $r1(d1)$*). However, this interpretation has been overruled by a subsequent decision (*$d2$*). The overruling is contrary to* $d1$*'s interpretation and indeed rebuts its validity (rule* $r2(d1)$*).*

$$\begin{aligned}
&\texttt{r0} \;:\; indecentExposure \;\Rightarrow\; violation\\
&\texttt{d1} \;:\; breastFeeding \;\Rightarrow\; indecentExposure\\
&\texttt{d2} \;:\; \Rightarrow\; overruled(decision(d1))\\
&\texttt{r1(x)} \;:\; decision(X) \;\Rightarrow\; \mathsf{usable}(X)\\
&\texttt{r2(x)} \;:\; \Rightarrow\; \mathsf{contr}(overrruled(decision(X)), \mathsf{usable}(X))
\end{aligned}$$

Arguments obtained from the knowledge base $K_p = breastFeeding,$ $decision(d1), decision(d2), \mathsf{usable}(r0), \mathsf{usable}(r1(X)), \mathsf{usable}(r2(X))$ *follow:*

A1: $breastFeeding$

A2 : $decision(d1)$

A3 : $decision(d2)$

A4 : $\textsf{usable}(r0)$

A5 : $\textsf{usable}(r1(d1))$

A6 : $\textsf{usable}(r1(d2))$

A7: $\textsf{usable}(r2(d1))$

A8: *A5,A2* $\Rightarrow$ $\textsf{usable}(d1)$

A9: *A6,A3* $\Rightarrow$ $\textsf{usable}(d2)$

A10: *A8,A1* $\Rightarrow$ $indecentExposure$

A11: *A9* $\Rightarrow$
$overrruled(decision(d1))$

A12: *A7* $\Rightarrow$
$\textsf{contr}(overrruled(decision(d1)), \textsf{usable}(d1))$

A13: *A4,A10* $\Rightarrow$ $violation$

CA11: *A12,A9* $\Rightarrow$
$overruled(decision(d1))$

Figure 6.2 shows the argumentation graph evaluated with grounded semantics. Let us shortly examine the arguments. Argument $A1$*-*$A3$ *affirm the basic facts in* K_p*. Arguments* $A4$*-*$A7$ *affirm the validity of the rule instances to be used in subsequent arguments. In particular* $A6$ *and* $A7$ *conclude for the validity of rules according to which if a decision is adopted then its content is valid.* $A8$ *and* $A9$ *use the latter rules for concluding for the validity of the content of* $d1$ *and* $d2$*.* $A10$ *uses the validity of* $d1$ *according to* $A8$ *to conclude, using* $A1$ *(breastfeeding), for indecent exposure.* $A11$ *uses* $A7$ *to conclude that* $d1$ *is overruled, while* $A12$ *concludes that* $d1$ *being overruled collides with its validity. Finally the two latter arguments are used to build an attack arguments (*$CA11$*) according to which* $d1$ *is rebutted by its overruling. As result, no violation is found (*$A13$*).*

In our previous example, we navigated through a problem involving past decisions and interpretations under the precedential approach, primarily employing reasoning about rules and conflicts. However, in that scenario, the canon was not explicitly modeled, and the element of preferences did not play a role. Now, we turn our attention to a more intricate scenario where competing canons and preferences are actively involved. This final example will illustrate the depth and

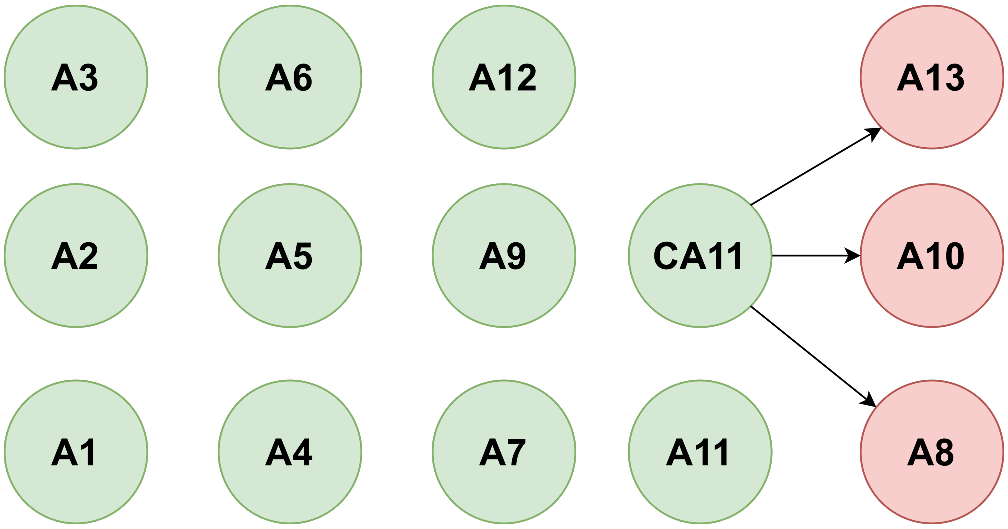


Figure 6.2: Argumentation framework from Example 14.

complexity our model can handle, showcasing its capability to engage with nuanced legal interpretation scenarios where multiple interpretative canons intersect and where preferences play a significant role in determining the outcome.

Example 15. *Interpretation - Unfair dismissal*

Nino Mezza, who worked as a software engineer for the British company Dif's & Ric, was unfairly dismissed after his complaints for the non-payment of overtime wages by the company. The tribunal has now to determine the amount of the compensatory award. The relevant piece of legislation is the Section 123(1) of the UK Employment Rights Act 1996, that reads: "The amount of the compensatory award shall be such amount as the tribunal considers just and equitable in all the circumstances having regard to the loss sustained by the complainant in consequence of the dismissal".

*The employer argued that the relevant section of the current UK legislation (p) only allows for the recovery of financial loss (i*1*) (ordinary meaning of the term loss). Conversely, the employee argued that an interpretation of this provision in the light of the original intent would grant him the recovery of losses other than financial, including also moral losses (i*2*). Ultimately, to determine whether the claimant should be compensated not only for his financial losses, but also for his moral harm, the scope of the term loss has to be determined. In this example we*

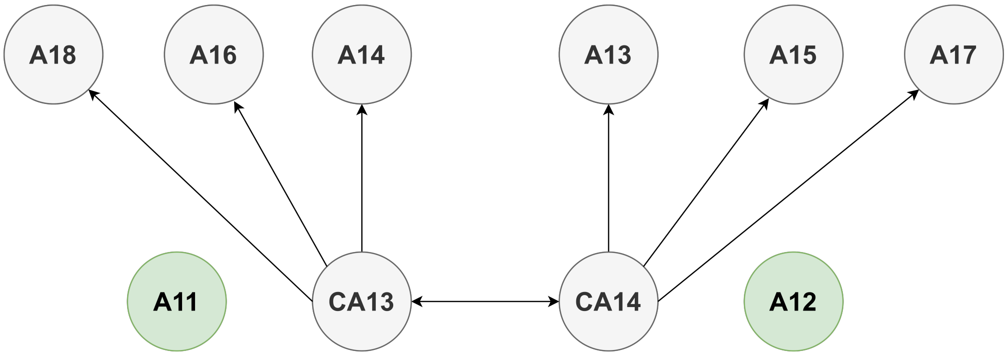


Figure 6.3: Argumentation framework from Example 15 with no priorities over available interpretations. Arguments $A0, \dots, A10$ – unconflicted premises and axioms – are omitted for space reasons.

will try to determine if the introduced meta-argumentation framework possesses all the features required to represent and automatically answer the above question given all the relevant knowledge.

Consider R_d as follows:

$\texttt{p:}\ unfairlyDismissed\texttt{,}\ loss \Rightarrow rightToCompensation$
$\texttt{i1:}\ financialLoss \Rightarrow loss$
$\texttt{i2:}\ moralLoss \Rightarrow loss$

$\texttt{r(Y,X):}\ provision(X)\texttt{,}\ interpretation(Y, X)$
$\quad \Rightarrow \mathsf{usable}(Y)$
$\texttt{c(X,Y,Z):}\ interpretation(X, Z)\texttt{,}\ interpretation(Y, Z)\texttt{,}\ Y \neq Y\texttt{,}$
$\quad \Rightarrow \mathsf{contr}(\mathsf{usable}(X), \mathsf{usable}(Y))$
$\texttt{s(X,Y,Z):}\ purposive(X, Z)\texttt{,}\ literal(Y, Z)$
$\quad \Rightarrow \mathsf{sup}(r(X, Z), r(Y, Z))$

the following set of ordinary premises K_d:

$provision(p1)$
$interpretation(i1, p)$
$interpretation(i2, p)$

$unfairlyDismissed$
$financialLoss$
$moralLoss$

and $K_p = \{\mathsf{usable}(s(X, Y, Z)), \mathsf{usable}(r(X, Y)), \mathsf{usable}(c(X, Y, Z)), \mathsf{usable}(p)\}$.

According to rule $r(Y, X)$ interpretations of legal provisions count as valid rules.

According to rule $c(X, Y, Z)$ *different interpretation of the same legal are contrary one to the other. According to rule* $s(X, Y, Z)$*, the ascriptions of validity (according to rule* $r(Y, X)$*) to a purposive interpretation prevail over ascriptions of validity to a literal interpretation. The knowledge base contains all the relevant facts: the existence of a main provision (*$provision(p)$*), the of available interpretations (*$interpretation(i1, p)$ *and* $interpretation(i2, p)$*), Nino's unfair dismissal (*$unfairlyDismissed$*) and a proof of the losses he sustained (*$financialLoss$ *and* $moralLoss$*).*

These are the arguments that we can build from the theory:

A0: $provision(p)$
A1: $interpretation(i1, p)$
A2: $interpretation(i2, p)$
A3: $unfairlyDismissed$
A4: $financialLoss$
A5: $moralLoss$

A6: $\mathsf{usable}(p)$
A7: $\mathsf{usable}(r(i1, p))$
A8: $\mathsf{usable}(r(i2, p))$
A9: $\mathsf{usable}(c(i1, i2, p))$
A10: $\mathsf{usable}(c(i2, i1, p))$

A11 : A9, A1, A2 $\Rightarrow$ $\mathsf{contr}(\mathsf{usable}(i1), \mathsf{usable}(i2))$
A12 : A10, A1, A2 $\Rightarrow$ $\mathsf{contr}(\mathsf{usable}(i2), \mathsf{usable}(i1))$
A13 : A7, A0, A1 $\Rightarrow$ $\mathsf{usable}(i1)$
A14 : A8, A0, A2 $\Rightarrow$ $\mathsf{usable}(i2)$
A15 : A13, A4 $\Rightarrow$ $loss$
A16 : A14, A5 $\Rightarrow$ $loss$
A17 : A0, A15 $\Rightarrow$ $rightToCompensation$
A18 : A0, A16 $\Rightarrow$ $rightToCompensation$
CA13 : A11, A7, A0, A1 $\Rightarrow$ $\mathsf{usable}(i1)$
CA14 : A12, A8, A0, A2 $\Rightarrow$ $\mathsf{usable}(i2)$

Figure 6.3 shows the resulting graph evaluated under grounded semantics (Last-Link ordering). The arguments standing for the validity of the two competing interpretations ($A13$ *and* $A14$*) attack each other through their respective Conflict Arguments (*$CA13$ *and* $CA14$*). Since the knowledge base does not contain any reason to prefer one interpretation over the other, it is not possible to solve their conflict and, consequently, the graph remains mostly undetermined.*

Let us now add to the system the knowledge on the interpretative canons orig-

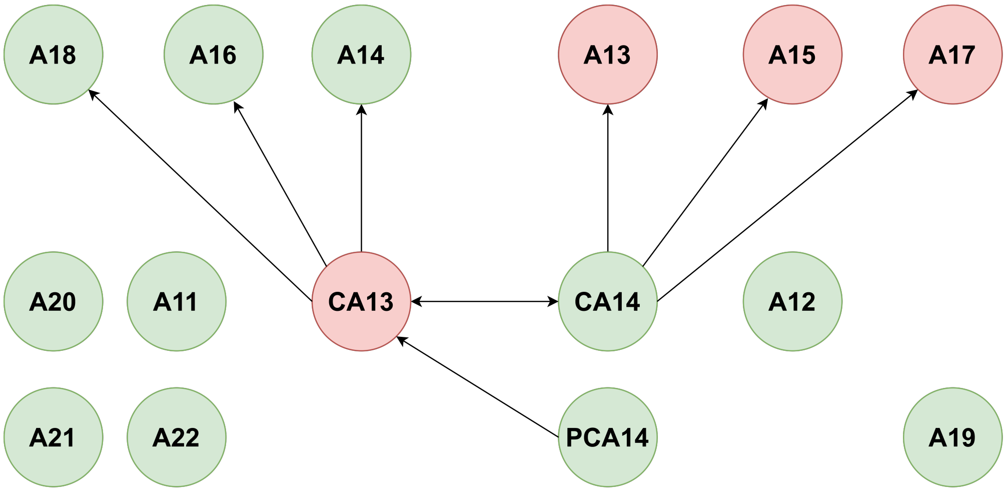


Figure 6.4: Argumentation framework from Example 15 with literal preferred over other interpretations. Arguments $A0, \dots, A10$ – unconflicted premises and axioms –, and attacks from $PCA14$ to $A13, A15, A17$ are omitted for space reasons.

inating the interpretations, namely a literal approach for $i1$ *and a purposive one for* $i2$*. Accordingly, we can define* $K' = K \cup \{literal(i1,p), purposive(i2,p)\}$*. The arguments that we can build using the new knowledge follow:*

$$\begin{aligned}
\mathtt{A19} &: \mathsf{usable}(s(i2,i1,p)) \\
\mathtt{A20} &: literal(i1,p) \\
\mathtt{A21} &: purposive(i2,p) \\
\mathtt{A22} &: \mathtt{A19}, \mathtt{A21}, \mathtt{A20} \Rightarrow \mathsf{sup}(r(i2,p), r(i1,p)) \\
\mathtt{PCA14} &: \mathtt{A22}, \mathtt{A12}, \mathtt{A8}, \mathtt{A0}, \mathtt{A2} \Rightarrow \mathsf{usable}(i2)
\end{aligned}$$

Figure 6.4 shows the resulting graph evaluated under grounded semantics (Last-Link ordering). The preference claimed by $A22$ *is used to build a Preference Argument based on* $CA14$*. The resulting argument (*$PCA14$*) defeats – without being defeated –* $CA13$*, i.e., the literal interpretation is found invalid for the benefit of the purposive interpretation of the term loss. As result, Nino is found eligible for the compensation of all his losses (*$A18$*).*

In conclusion, the examples presented throughout this section effectively demonstrate the robustness and versatility of our model. Each scenario, ranging from the nuanced interpretation of legal definitions to the complex balancing of competing legal principles, has highlighted the model's ability to navigate and resolve intricate legal issues. Indeed, these examples not only illustrate the model's capacity to engage with and apply various interpretative canons but also showcase its proficiency in managing preferences and conflicts within legal reasoning. This demonstration reinforces the model's potential as a valuable asset in the field of legal AI, proving its efficacy in tackling the multifaceted challenges of legal reasoning.

6.2 Computable Law via Arg2P

In this section, we aim to provide a comprehensive demonstration of the Arg2P framework within the context of computable law. The examples we present will not only showcase the diverse features of the framework that are applicable to intelligent systems in general, but will also focus specifically on illustrating how Arg2P effectively meets a range of critical legal requirements.

The general features of the Arg2P framework that will be highlighted include system transparency and explainability, customization and integration capabilities, as well as its proficiency in non-monotonic reasoning and defeasibility. These features are fundamental to intelligent systems, offering enhanced interpretability, adaptability to various contexts, and the ability to handle changing and conflicting information.

In addition to these general features, our demonstration will particularly emphasize how Arg2P satisfies several key legal requirements in the treatment of legal knowledge. These requirements, as discussed in [Gordon et al., 2009], include:

- *Rules form and strict semantic*, ensuring each legal rule is formulated with precise and unambiguous semantics.

- *Support for knowledge defeasibility through conflicts and exclusionary rules*, i.e., the ability to adapt to new information and resolve conflicts between established rules.

- *No contraposition in defeasible rules*, i.e., the falseness of the conclusion of a defeasible rule does not lead to any assumption about the validity of the rule's premises.

- *Attribution of value over rules through preferences*, i.e., prioritizing certain legal principles or rules over others in cases of conflict or ambiguity.

- *Deontic effects*, i.e., handling normative concepts such as obligations, permissions, and prohibitions.

Furthermore, to provide a more rounded view of the framework's capabilities in computable law, we will also consider additional legal requirements such as the ability of the system to understand and react to the legal context in which it operates, and ensuring that the framework operates within ethical boundaries and garners trust from its users, especially in sensitive legal applications.

Through these examples, our goal is to clearly demonstrate the efficacy and suitability of the Arg2P framework for computable law applications, showcasing its ability to meet the sophisticated demands of legal reasoning and knowledge management in intelligent systems.

In this section, we will explore two distinct application areas. Firstly, some examples in the fields of computable law for autonomous vehicles are discussed to show the effectiveness of Arg2P in distributing intelligence and reasoning capabilities over AI applications. In this scenario, involving autonomous cars and relative legal computation, vehicles are capable of communicating with each other and with the road infrastructure. Cities and roads are suitably enriched with sensors and virtual traffic signs able to dynamically interact with cars to provide information and supervision. Accordingly, self-driving cars need to *(i)* exhibit some degree of intelligence for making autonomous decisions; they need to *(ii)* interact with the context that surrounds them, *(iii)* have humans in the loop, *(iv)* respond to the legal setting characterising the environment and the society, and *(v)* offer explanations when required—e.g., in case of accidents to determine causes and responsibilities.

Following this, the second area will shift our attention to the use of the Arg2P framework within the context of criminal justice, showcasing its application and effectiveness in this complex legal environment.

6.2.1 Autonomous cars & Legal Reasoning

First of all, we consider a very simple scenario in the context of autonomous cars: a road equipped with two traffic lights, one for the vehicles and one for the pedestrians. The goal of the system is to autonomously manage intersections accordingly to traffic light indications. A complication should be taken into account, that is: an authorised vehicle could – during emergencies – ignore traffic light prescriptions. In that case, other vehicles must leave the way clear for the authorised vehicle.

Listing 6.1 encodes the rules in the Arg2P system, whereas Listing 6.2 encodes the corresponding arguments.

Listing 6.1: Example 1 theory

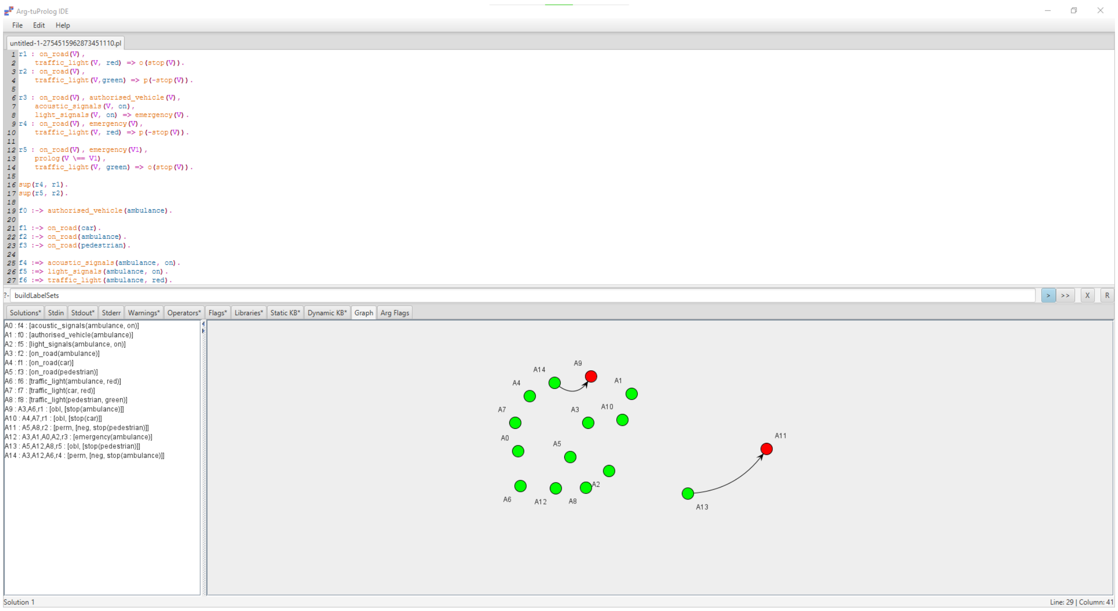
```
r1 : on_road(V), traffic_light(V, red) => o(stop(V)).
r2 : on_road(V), traffic_light(V,green) => p(-stop(V)).
r3 : on_road(V), authorised_vehicle(V), acoustic_signals(V, on), light_signals(V, on)=> emergency(V).
r4 : on_road(V), emergency(V),traffic_light(V, red) => p(-stop(V)).
r5 : on_road(V), emergency(V1), prolog(V \== V1), traffic_light(V, green) => o(stop(V)).

sup(r4, r1).
sup(r5, r2).

f0 :-> authorised_vehicle(ambulance).
f1 :-> on_road(car).
f2 :-> on_road(ambulance).
f3 :-> on_road(pedestrian).
f4 :=> acoustic_signals(ambulance, on).
f5 :=> light_signals(ambulance, on).
f6 :=> traffic_light(ambulance, red).
f7 :=> traffic_light(car, red).
f8 :=> traffic_light(pedestrian, green).
```

Listing 6.2: Arguments from Listing 6.1

```
A0 : f4 ⟹ acoustic_signals(ambulance, on) A8 : f8 ⟹ traffic_light(pedestrian, green)
A1 : f0 ⟹ authorised_vehicle(ambulance)  A9 : A3,A6,r1 ⟹ o(stop(ambulance))
A2 : f5 ⟹ light_signals(ambulance, on)   A10 : A4,A7,r1 ⟹ o(stop(car))
A3 : f2 ⟹ on_road(ambulance)     A11 : A5,A8,r2 ⟹ p(-stop(pedestrian))
A4 : f1 ⟹ on_road(car)           A12 : A3,A1,A0,A2,r3 ⟹ emergency(ambulance)
A5 : f3 ⟹ on_road(pedestrian)]       A13 : A5,A12,A8,r5 ⟹ o(stop(pedestrian))
A6 : f6 ⟹ traffic_light(ambulance, red)  A14 : A3,A12,A6,r4 ⟹ p(-stop(ambulance))
A7 : f7 ⟹ traffic_light(car, red)
```

Rules `r1` and `r2` represent fundamental constraints: if the traffic light is red, road users – e.g. pedestrians, cars, etc. – have to stop; otherwise they can proceed. Rules `r3` and `r4` model the concept of a vehicle in an emergency, giving it permission to proceed even if the light is red. Rule `r5` imposes other road users the obligation to stop if aware of another vehicle in an emergency state.[1] Finally,

[1] The `prolog(...)` term is a special Arg2P notation that allows using as a premise a Prolog expression. In this case, it is used to avoid the unification between the variable `Z` and `Y`, which



```
r1 : on_road(V) ,
    traffic_light(V, red) => o(stop(V)).
r2 : on_road(V) ,
    traffic_light(V,green) => p(-stop(V)).

r3 : on_road(V) , authorised_vehicle(V),
    acoustic_signals(V, on),
    light_signals(V, on) => emergency(V).
r4 : on_road(V) , emergency(V),
    traffic_light(V, red) => p(-stop(V)).

r5 : on_road(V) , emergency(V1) ,
    prolog(V \== V1) ,
    traffic_light(V, green) => o(stop(V)).

sup(r4, r1).
sup(r5, r2).

f0 :-> authorised_vehicle(ambulance).

f1 :-> on_road(car).
f2 :-> on_road(ambulance).
f3 :-> on_road(pedestrian).

f4 :=> acoustic_signals(ambulance, on).
f5 :=> light_signals(ambulance, on).
f6 :=> traffic_light(ambulance, red).
```

```
?- buildLabelSets
```

```
A0 : f4 : [acoustic_signals(ambulance, on)]
A1 : f0 : [authorised_vehicle(ambulance)]
A2 : f5 : [light_signals(ambulance, on)]
A3 : f2 : [on_road(ambulance)]
A4 : f1 : [on_road(car)]
A5 : f3 : [on_road(pedestrian)]
A6 : f6 : [traffic_light(ambulance, red)]
A7 : f7 : [traffic_light(car, red)]
A8 : f8 : [traffic_light(pedestrian, green)]
A9 : A3,A6,r1 : [obl, [stop(ambulance)]]
A10 : A4,A7,r1 : [obl, [stop(car)]]
A11 : A5,A8,r2 : [perm, [neg, stop(pedestrian)]]
A12 : A3,A1,A0,A2,r3 : [emergency(ambulance)]
A13 : A5,A12,A8,r5 : [obl, [stop(pedestrian)]]
A14 : A3,A12,A6,r4 : [perm, [neg, stop(ambulance)]]
```

Figure 6.5: Example 1 grounded argumentation graph in Arg2P IDE

two preferences are specified—the first on the rule `r4` over `r1` and the second on `r5` over `r2`. These preferences assign a higher priority to emergency situations – `r4` and `r5` – over ordinary ones—`r1` and `r2`. Facts from `f0` to `f8` depict a situation in which there are three users on road: a car, an ambulance and a pedestrian. The ambulance has its acoustic and light indicators on—stating an emergency situation. The traffic light is red both for the ambulance and the car, and green for the pedestrian.

With respect to permissions and obligations, the only argument that can be built about the car is A_{10}, declaring the obligation to stop—A_{10} via `r1`. For the pedestrian and the ambulance, the situation is more faceted. In both cases, two conflicting arguments can be built: one stating the permission to proceed for the pedestrian and for the ambulance – A_{11} and A_{14} respectively – and one stating the obligation to stop—A_{13} and A_9 respectively. These arguments rebut each other: yet, taking into account the preferences over `r4` and `r5`, the acceptability of the arguments stating the obligation to stop for the pedestrian and the permission to cross for the ambulance can be established (Figure 6.5). Essential to this outcome is the emergency state of the ambulance (A_{12}): if it were not possible to prove the

would lead to emergency vehicles having the obligation to stop at their own passage.

emergency of the situation – it is required for an authorised vehicle to have both acoustic and light signals on –, then the vehicle would have to stop (A_9) leaving free the pedestrian to proceed (A_{11}).

Listing 6.3: Example 2 theory

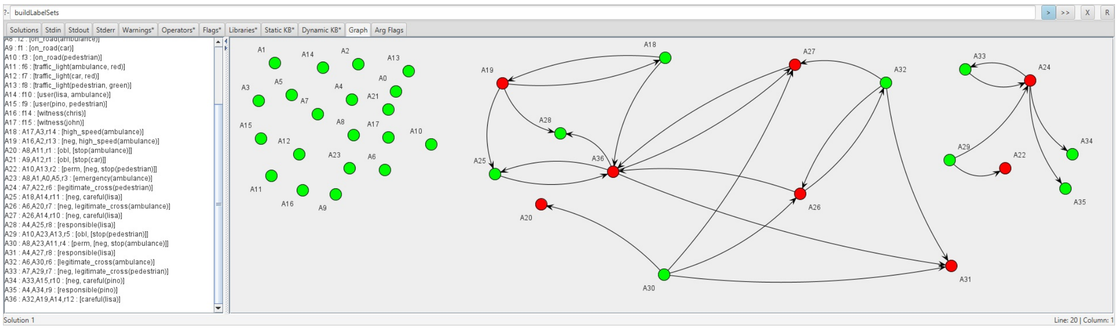
```
r6 : -stop(V), p(-stop(V)) => legitimate_cross(V).
r7 : -stop(V), o(stop(V)) => -legitimate_cross(V).
r8 : harms(P1, P2), -careful(P1) => responsible(P1).
r9 : harms(P1, P2), -careful(P2) => responsible(P2).
r10 : -legitimate_cross(V),  user(P, V) => -careful(P).
r11 : high_speed(V), user(P, V)  => -careful(P).
r12 : legitimate_cross(V), -high_speed(V), user(P, V)  => careful(P).
r13 : witness(X), claim(X, low_speed(V)) => -high_speed(V).
r14 : witness(X), claim(X, high_speed(V)) => high_speed(V).

bp(careful(P)).

f9 :-> user(pino, pedestrian).
f10 :-> user(lisa, ambulance).
f11 :-> -stop(ambulance).
f12 :-> -stop(pedestrian).
f13 :-> harms(lisa, pino).
f14 :-> witness(chris).
f15 :-> witness(john).
f16 :=> claim(chris, low_speed(ambulance)).
f17 :=> claim(john, high_speed(ambulance)).
```

Listing 6.4: Arguments from Listing 6.1 and 6.3

```
A0 : f4  ⟹  acoustic_signals(ambulance, on) A19 : A16,A2,r13  ⟹  -high_speed(ambulance)
A1 : f0  ⟹  authorised_vehicle(ambulance)  A20 : A8,A11,r1  ⟹  o(stop(ambulance))
A2 : f16  ⟹  claim(chris, low_speed(ambulance)) A21 : A9,A12,r1  ⟹  o(stop(car))
A3 : f17  ⟹  claim(john, high_speed(ambulance)) A22 : A10,A13,r2  ⟹  p(-stop(pedestrian))
A4 : f13  ⟹  harms(lisa, pino)     A23 : A8,A1,A0,A5,r3  ⟹  emergency(ambulance)
A5 : f5  ⟹  light_signals(ambulance, on)   A24 : A7,A22,r6  ⟹  legitimate_cross(pedestrian)
A6 : f11  ⟹  -stop(ambulance)      A25 : A18,A14,r11  ⟹  -careful(lisa)
A7 : f12  ⟹  -stop(pedestrian)     A26 : A6,A20,r7  ⟹  -legitimate_cross(ambulance)
A8 : f2  ⟹  on_road(ambulance)     A27 : A26,A14,r10  ⟹  -careful(lisa)
A9 : f1  ⟹  on_road(car)           A28 : A4,A25,r8  ⟹  responsible(lisa)
A10 : f3  ⟹  on_road(pedestrian)       A29 : A10,A23,A13,r5  ⟹  o(stop(pedestrian))
A11 : f6  ⟹  traffic_light(ambulance, red)  A30 : A8,A23,A11,r4  ⟹  p(-stop(ambulance))
A12 : f7  ⟹  traffic_light(car, red)   A31 : A4,A27,r8  ⟹  responsible(lisa)
A13 : f8  ⟹  traffic_light(pedestrian, green) A32 : A6,A30,r6  ⟹  legitimate_cross(ambulance)
A14 : f10  ⟹  user(lisa, ambulance)    A33 : A7,A29,r7  ⟹  -legitimate_cross(pedestrian)
A15 : f9  ⟹  user(pino, pedestrian)    A34 : A33,A15,r10  ⟹  -careful(pino)
A16 : f14  ⟹  witness(chris)       A35 : A4,A34,r9  ⟹  responsible(pino)
A17 : f15  ⟹  witness(john)        A36 : A32,A19,A14,r12  ⟹  careful(lisa)
A18 : A17,A3,r14  ⟹  high_speed(ambulance)
```

The focus in the previous example is on the plane of duties, i.e., automatic reasoning aimed at defining what is permitted / prohibited in the contingent situation. Let us take a step further.

The ambulance, driven by Lisa, has permission to move despite the red light due to an emergency situation; the pedestrian, Pino, has the obligation to stop. Let us imagine that Pino, despite the prohibition to proceed, keeps on crossing.



```
A8 : f2 : [on_road(ambulance)]
A9 : f1 : [on_road(car)]
A10 : f3 : [on_road(pedestrian)]
A11 : f6 : [traffic_light(ambulance, red)]
A12 : f7 : [traffic_light(car, red)]
A13 : f8 : [traffic_light(pedestrian, green)]
A14 : f10 : [user(lisa, ambulance)]
A15 : f9 : [user(pino, pedestrian)]
A16 : f14 : [witness(chris)]
A17 : f15 : [witness(john)]
A18 : A17,A3,r14 : [high_speed(ambulance)]
A19 : A16,A2,r13 : [neg, high_speed(ambulance)]
A20 : A8,A11,r1 : [obl, [stop(ambulance)]]
A21 : A9,A12,r1 : [obl, [stop(car)]]
A22 : A10,A13,r2 : [perm, [neg, stop(pedestrian)]]
A23 : A8,A1,A0,A5,r3 : [emergency(ambulance)]
A24 : A7,A22,r6 : [legitimate_cross(pedestrian)]
A25 : A18,A14,r11 : [neg, careful(lisa)]
A26 : A6,A20,r7 : [neg, legitimate_cross(ambulance)]
A27 : A26,A14,r10 : [neg, careful(lisa)]
A28 : A4,A25,r8 : [responsible(lisa)]
A29 : A10,A23,A13,r5 : [obl, [stop(pedestrian)]]
A30 : A8,A23,A11,r4 : [perm, [neg, stop(ambulance)]]
A31 : A4,A27,r8 : [responsible(lisa)]
A32 : A6,A30,r6 : [legitimate_cross(ambulance)]
A33 : A7,A29,r7 : [neg, legitimate_cross(pedestrian)]
A34 : A33,A15,r10 : [neg, careful(pino)]
A35 : A4,A34,r9 : [responsible(pino)]
A36 : A32,A19,A14,r12 : [careful(lisa)]
```

Figure 6.6: Example 2 bp labelling in Arg2P IDE

The result has been an accident in which Pino has been harmed by the ambulance, which failed to see him and has not stopped its run. The purpose here is to find the responsibilities of the parties in the accident.

For instance, let us suppose the case is under the Italian jurisdiction, so that the Italian law is to be applied. According to Italian law, responsibility in an accident is based on the concept of carefulness. Both Lisa and Pino have to prove that they were careful (i.e., prudent) and acted according to the law. If they fail to prove such facts, they are considered responsible for the event, i.e., they both have the burden of persuasion on carefulness.

In the following, we discuss how the Arg2P reasoner enables to deal with that sort of situation. Listing 6.3 shows a possible representation of these rules in Arg2P.

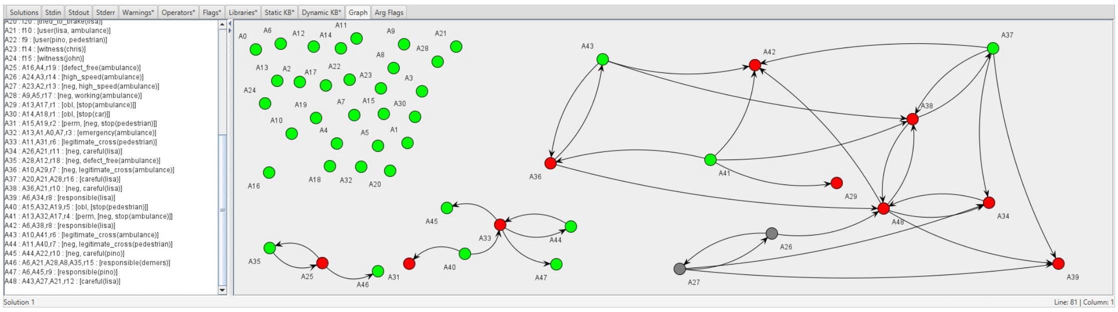
Rules `r6` and `r7` define the concepts of permitted and prohibited crossing: if a road-user has to stop but doesn't stop, he has to be considered responsible for causing accidents and related damages. Rules `r8` and `r9` encode the notion of responsibility in an accident, bound to the carefulness of the road-users involved. Rules `r10`, `r11` and `r12` define the carefulness of a subject. Accordingly, a road user can be considered careful if the crossing is permitted and his/her speed is not excessive. Otherwise, he/she is considered imprudent. Finally, rules `r13` and `r14` state the speed of a road user based on the testimonials of any witnesses. The `bp(careful(X))` notation allocates the burden of persuasion on the carefulness of each party, i.e., it is required to the parties to provide evidence for that. If they fail to meet the burden, carefulness arguments are rejected. Facts from `f9` to `f17` contain the available knowledge: both Pino and Lisa did not stop at the crossing and, consequently, Lisa harmed Pino. There are two witnesses, John and Chris, the first claiming that the ambulance driven by Lisa was maintaining the proper speed, and the other claiming that she was proceeding at high speed.

With respect to the grounded semantic, the argument for Pino's responsibility (A_{34} via `r9`) is accepted – he is guilty of its forbidden crossing (A_{35} via `r10`) – and one argument claiming Lisa's responsibility is rejected (A_{31}). Indeed, the argument for Lisa's uncarefulness (A_{27} via `r10`) is based on the premise of Lisa's forbidden crossing (A_{26} via `r7`) that is defeated by the legitimacy of her action (A_{24} via `r6` stating the case of emergency). Lisa's responsibilities in the accident remain uncertain due to the two contradicting witnesses – rebutting each other – i.e., the system can derive both Lisa being careful (`r12`) and not being careful (`r11`). So, Lisa's responsibilities are left undecided. The grounded semantics does not provide the legally correct answer.

In the case at hand, indeed, a semantic related to the burden of persuasion needs to be considered. The execution under the bp semantics [Calegari et al., 2021d] (Figure 6.6) concludes for the responsibility of the ambulance driver in the event. The uncertainty on Lisa's carefulness is considered as a failure to meet the burden of persuasion on the claim `careful(lisa)`. Consequently, the argument supporting this claim (A_{36}) is rejected, leaving space for the admissibility of the conflicting arguments.

Listing 6.5: Example 3 theory

```
r15 : harms(P1, P2), user(P1, V), -working(V), manufacturer(M, V), -defect_free(V) => responsible(M).
r16 : tried_to_brake(P), user(P, V),  -working(V) => careful(P).
r17 : mechanic(M), claim(M, defect(V)) => -working(V).
r18 : -working(V), new(V) => -defect_free(V).
r19 : production_manager(P), claim(P, test_ok(V)) => defect_free(V).
r20 : test_doc_ok(V) => undercut(r18).
sup(r16, r11).
bp(defect_free(V)).
f19 :-> manufacturer(demers , ambulance).
f20 :=> tried_to_brake(lisa).
f21 :-> mechanic(paul).
f22 :=> claim(paul, defect(ambulance)).
f23 :-> new(ambulance).
f24 :-> production_manager(mike).
f25 :=> claim(mike, test_ok(ambulance)).
```



```
A20 : f20 : [tried_to_brake(lisa)]
A21 : f10 : [user(lisa, ambulance)]
A22 : f9 : [user(pino, pedestrian)]
A23 : f14 : [witness(chris)]
A24 : f15 : [witness(john)]
A25 : A16,A4,r19 : [defect_free(ambulance)]
A26 : A24,A3,r14 : [high_speed(ambulance)]
A27 : A23,A2,r13 : [neg, high_speed(ambulance)]
A28 : A9,A5,r17 : [neg, working(ambulance)]
A29 : A13,A17,r1 : [obl, [stop(ambulance)]]
A30 : A14,A18,r1 : [obl, [stop(car)]]
A31 : A15,A19,r2 : [perm, [neg, stop(pedestrian)]]
A32 : A13,A1,A0,A7,r3 : [emergency(ambulance)]
A33 : A11,A31,r6 : [legitimate_cross(pedestrian)]
A34 : A26,A21,r11 : [neg, careful(lisa)]
A35 : A28,A12,r18 : [neg, defect_free(ambulance)]
A36 : A10,A29,r7 : [neg, legitimate_cross(ambulance)]
A37 : A20,A21,A28,r16 : [careful(lisa)]
A38 : A36,A21,r10 : [neg, careful(lisa)]
A39 : A6,A34,r8 : [responsible(lisa)]
A40 : A15,A32,A19,r5 : [obl, [stop(pedestrian)]]
A41 : A13,A32,A17,r4 : [perm, [neg, stop(ambulance)]]
A42 : A6,A38,r8 : [responsible(lisa)]
A43 : A10,A41,r6 : [legitimate_cross(ambulance)]
A44 : A11,A40,r7 : [neg, legitimate_cross(pedestrian)]
A45 : A44,A22,r10 : [neg, careful(pino)]
A46 : A6,A21,A28,A8,A35,r15 : [responsible(demers)]
A47 : A6,A45,r9 : [responsible(pino)]
A48 : A43,A27,A21,r12 : [careful(lisa)]
```

Figure 6.7: Example 3 bp labelling in Arg2P IDE

Listing 6.6: Arguments from Listing 6.1, 6.3 and 6.5

```
A0 : f4 ⟹ acoustic_signals(ambulance, on) A25 : A16,A4,r19 ⟹ defect_free(ambulance)
A1 : f0 ⟹ authorised_vehicle(ambulance)  A26 : A24,A3,r14 ⟹ high_speed(ambulance)
A2 : f16 ⟹ claim(chris, low_speed(ambulance)) A27 : A23,A2,r13 ⟹ -high_speed(ambulance)
A3 : f17 ⟹ claim(john, high_speed(ambulance)) A28 : A9,A5,r17 ⟹ -working(ambulance)
A4 : f25 ⟹ claim(mike, test_ok(ambulance)) A29 : A13,A17,r1 ⟹ o(stop(ambulance))
A5 : f22 ⟹ claim(paul, defect(ambulance)) A30 : A14,A18,r1 ⟹ o(stop(car))
A6 : f13 ⟹ harms(lisa, pino)      A31 : A15,A19,r2 ⟹ p(-stop(pedestrian))
A7 : f5 ⟹ light_signals(ambulance, on)   A32 : A13,A1,A0,A7,r3 ⟹ emergency(ambulance)
A8 : f19 ⟹ manufacturer(demers, ambulance) A33 : A11,A31,r6 ⟹ legitimate_cross(pedestrian)
A9 : f21 ⟹ mechanic(paul)        A34 : A26,A21,r11 ⟹ -careful(lisa)
A10 : f11 ⟹ -stop(ambulance)     A35 : A28,A12,r18 ⟹ -defect_free(ambulance)
A11 : f12 ⟹ -stop(pedestrian)        A36 : A10,A29,r7 ⟹ -legitimate_cross(ambulance)
A12 : f23 ⟹ new(ambulance)       A37 : A20,A21,A28,r16 ⟹ careful(lisa)
A13 : f2 ⟹ on_road(ambulance)        A38 : A36,A21,r10 ⟹ -careful(lisa)
A14 : f1 ⟹ on_road(car)          A39 : A6,A34,r8 ⟹ responsible(lisa)
A15 : f3 ⟹ on_road(pedestrian)       A40 : A15,A32,A19,r5 ⟹ o(stop(pedestrian))
A16 : f24 ⟹ production_manager(mike) A41 : A13,A32,A17,r4 ⟹ p(-stop(ambulance))
A17 : f6 ⟹ traffic_light(ambulance, red) A42 : A6,A38,r8 ⟹ responsible(lisa)
A18 : f7 ⟹ traffic_light(car, red)   A43 : A10,A41,r6 ⟹ legitimate_cross(ambulance)
A19 : f8 ⟹ traffic_light(pedestrian, green) A44 : A11,A40,r7 ⟹ -legitimate_cross(pedestrian)
A20 : f20 ⟹ tried_to_brake(lisa)     A45 : A44,A22,r10 ⟹ -careful(pino)
A21 : f10 ⟹ user(lisa, ambulance)    A46 : A6,A21,A28,A8,A35,r15 ⟹ responsible(demers)
A22 : f9 ⟹ user(pino, pedestrian)    A47 : A6,A45,r9 ⟹ responsible(pino)
A23 : f14 ⟹ witness(chris)       A48 : A43,A27,A21,r12 ⟹ careful(lisa)
A24 : f15 ⟹ witness(john)
```

6.2.2 More on legal reasoning

Let us extend the above-discussed example in which Lisa, the ambulance driver, and Pino, the pedestrian, were both considered responsible for the accident on the basis of the available knowledge. Lisa now declares that she tried to stop the ambulance, but the brake did not work. The ambulance is then sent to a mechanic, who states that, even if the ambulance is new, there is a problem with the brake system. In that case, the manufacturer is called to prove that the ambulance was not defective when delivered—i.e., the burden of proof on the adequacy of the vehicle is on the manufacturer.

At this stage, the discovery of a defect in the ambulance would lead to the discarding of Lisa's responsibility. Moreover, if the manufacturer fails to meet his burden, it would share the responsibilities of the accident.

Listing 6.5 shows a possible Arg2P encoding of the knowledge. Rule `r15` concludes the responsibility of the manufacturer in the case a malfunctioning is found on the vehicle and it is proved that there is a defect.

Rule `r16` infer the carefulness of the driver if a defect is found on the vehicle (for instance on the brake mechanism). The preference `sup(r16, r11)` states that in case of a defect carefulness should be inferred even if high speed has been detected. Rule `r17` states the evidence of a vehicle malfunctioning on the base of a mechanic declaration. Rules `r18`, `r19` and `r20` state the conditions for deducing in which cases the vehicle can be considered defect-free. The statement `bp(defect-free(X))` enforces the obligation for the manufacturer to prove its adherence to the regulations.

Facts `f19`–`f25` depict the above scenario: Paul the mechanic has found a problem in the brake system even if the ambulance is new. However, Mike, the production officer of the ambulance manufacturer, declares that every vehicle is deeply tested before the delivery and the vehicle at hand has been tested. Anyway, there is no trace of documentation.

The results of the evaluation of this scenario according to the bp semantics can be summarised as follows. On the one hand, Lisa is free from every responsibility in the accident since her prudence is correctly proved. Arguments A_{48} and A_{37} built on `r11` and `r16` defeat the A_{34} built on `r11` and consequently the one concluding her responsibility (A_{42} via `r8`) and the burden on carefulness can be considered satisfied. On the other hand, the manufacturer is found responsible for the accident (A_{46}). Indeed, arguments built on `r18` and `r19` – A_{35} and A_{25} respectively – rebut one other leading to a state of uncertainty. Hence, the burden is not satisfied, and the argument for the defect-free ambulance is rejected. Accordingly, the argument concluding the manufacturer's responsibility in the event is accepted.

The examples presented thus far highlight Arg2P as a robust framework capable of non-monotonic reasoning, which enhances system transparency and supports extensive customization for diverse legal applications. These instances align with the specific requirements of legal computation, showcasing Arg2P's effectiveness in meeting the complex needs of the legal domain, emphasizing its adaptability and proficiency in legal reasoning and computational challenges.

Our exploration, especially illustrated in the third example (Subsection 6.2.2), demonstrates how Arg2P's non-monotonic reasoning capability is ideally suited for legal scenarios, where knowledge is often unstable and subject to change. This adaptability is crucial in legal contexts, allowing for the accommodation of new information and the adjustment of legal interpretations as needed. Moreover, Arg2P fulfills several legal-specific requirements as outlined in [Gordon et al., 2009], particularly in terms of handling legal knowledge. This includes maintaining strict semantic forms of rules, supporting knowledge defeasibility with mechanisms to address conflicts, assigning priorities to rules based on preferences, and effectively managing deontic effects in legal reasoning.

From a different perspective, the intrinsic interpretability of the argumentation models used in Arg2P, as exemplified in the autonomous driving scenario (Subsection 6.2.1), contributes significantly to the system's transparency and explainability. This feature is vital in legal systems, where understanding the rationale behind decisions is crucial for accountability and trust. In scenarios like autonomous driving, where decisions are made based on sensor data, the ability of Arg2P to provide clear and understandable decision-making processes is of paramount importance.

Finally, the customizability of the Arg2P framework, as highlighted in Example 6.3, is particularly beneficial in the legal domain. Its modular architecture allows for seamless integration with other AI techniques and the addition of new semantics, like the one dealing with burden of persuasion [Calegari et al., 2021d]. This flexibility enables the framework to be tailored to specific legal requirements and scenarios, such as demonstrating compliance with safety regulations in autonomous vehicles.

6.2.3 Arg2P for Conformity Assessment of EU Regulations

The CrossJustice project studies the rights of defendants in criminal matters according to the laws of several EU Member States and provides a decision support system, accessible by professionals and citizens alike. The system provides assessments concerning specific cases. Moreover, it determines the level of harmonisation of national legislations, namely the extent to which national legal frameworks and regulatory acts linked are in line with the EU acquis and relevant legislative acts of the European Union.

The aim is to help legal practitioners in their daily activities, through a platform that supports interoperability and communication between the several legislative measures that single Member States have adopted, showing how these laws interact, and their compliance with the EU directives.

The main target of the project is the creation of a rule-based expert system grounded on a computable representation of the European directives[2] related to the rights of suspect and accused persons in criminal proceedings and the national articles concerning the same subject matter. The system shall inform the user of all the applicable rights according to both European and national acts.

All articles in such Directives and the relevant portions of national transpositions have been represented in Prolog. The legal analysis, carried out by expert lawyers, has often involved the interpretation of complex legal rules, and the reconstruction of the dependencies between norms. The main advantage of such a symbolic representation lies in its understandability, both to human programmers and to legal experts, and on the possibility to trace the reasoning processes.

The relation between directives and national laws presents a high level of complexity. In fact, differently from EU regulations, directives are meant to provide only a baseline, but each Member State is free to decide how to implement directives into national laws. As single Member States have a limited discretionary power in implementing these provisions, they can introduce discrepancies that may give rise to a wide spectrum of legal issues.

Legal professionals, besides verifying whether a particular right is recognised under a national jurisdiction, are also interested in the conditions under which that right is granted, and on the measures available for its protection. The aim of the system is thus twofold. First, it shall provide an answer regarding the recognition of such rights and their applicability in the national legal systems. Second, it shall verify the relation between the directive and the national implementation: it will identify where differences lie and highlight those cases in which the departure from

[2] Directive 2016/343, Directive 2010/64, Directive 2016/800, Directive 2016/1919, Directive 2012/13, Directive 2013/48

the directive entails a violation of it.

In this section, we will use an example to demonstrate how the Arg2P tool can highlight explicit differences between EU and national laws. In particular, it will focus on the consequences that a different definition given by a national legislator can have on its applicability, compared to those provided by EU law. It shall also show a visual representation of the conformity between the national legislation and the European Directive.

First, let us introduce the facts of the case in example. For simplicity and clarity's sake, we will only take into consideration the rules which are relevant to the case.

One of the points of contention between the Polish national law and the European Directives concerns the notion of 'suspect' ('*podejrzany*'). According to Polish law, only a person formally charged with a crime during a criminal investigation is granted a status of *podejrzany*. This charging decision, however, is only preliminary and not determined by a court. Instead, when an individual has been arrested or searched but not yet formally charged with a crime, the Polish law provides a separate name i.e. *osoba podejrzana* (translated directly as 'suspected person') and does not provide for her the same protection that the *podejrzany* is normally provided with. This term *osoba podejrzana*, is not present in the Polish translation of the Directive, since from the perspective of EU law an individual in such condition would fall under the definition of suspect.

The filing of the case with a court is done at a later stage, namely at the end of the criminal investigation, in the form of indictment which changes the person status from *podejrzany* to *oskarżony*—a party to court proceedings with all rights available to him or her at this stage of the criminal process.

We shall thus focus on a situation that emphasises this difference between the national law and EU directive, and showcase how logic programming and argumentation can lead the user to this conclusion while providing a sufficient explanation.

In order to demonstrate the full scope of our argumentation framework, we shall verify whether both the Directive and the Polish law recognise the same rights in an example case. We shall thus assume that a person has engaged in criminal activity related to an offence punishable by law, in Poland. During the investigation, the police deem it necessary to interrogate the accused and therefore informs him/her of the accusations that have been levelled. The defendant now decides to ask for the intervention of a legal counsel and interrogates the decision-support system for his or her rights.

Listing 6.7: Directive 2016/800 Prolog transposition

```
% r1
has_right(article4_a_iii, PersonId, right_to_information,
   privacy) :-
    person_status(PersonId, accused),
    user_fact(person_made_aware(PersonId, person_status)).

% r2
has_right(article6_3_a, PersonId, right_to_access_lawyer,
   questioning) :-
    person_status(PersonId, accused),
    user_fact(proceeding_matter(PersonId, questioning)).

% r3
person_status(PersonId, accused) :-
    user_fact(person_made_aware(PersonId, charge)).
```

Listing 6.7 illustrates the rules that apply in the case at hand, according to the Directive 2016/800 [3].

Rule `r1` illustrates the transposition of Article 4, paragraph 1, letter a(iii), Directive 2016/800. It states that a person has the right to be informed, as soon as he/she has been made aware of being accused of a crime, of the right to protection of privacy. Rule `r2` presents the transposition of Article 6, paragraph 3, letter a, of the Directive 2016/800, which states that a person has the right to be assisted by a lawyer without undue delay after being summoned for questioning. Finally, rule `r3` states that a person who, according to the Directive, has been made aware of charges against him/her is granted the status of accused. It is important to state that, nowhere in the Directive, this is explicitly stated. This rule (and a similar

[3] For the purpose of this case we shall ignore the fact that this Directive applies only to children

one later on) is an explication of the meaning of the term ‘suspect’ (and later on ‘accused’) as used in the context of the EU law.

Listing 6.8: Directive 2016/800’s Polish transposition in Prolog

```
% r4
has_right(art301, PersonId, right_to_access_lawyer,
   interrogation) :-
    person_status(PersonId, suspect),
    user_fact(proceeding_matter(PersonId, interrogation)),
    user_fact(person_request_submitted(PersonId,
       defence_counsel)).

% r5
person_status(PersonId, suspect) :-
    user_fact(person_made_aware(PersonId, charge)),
    \+ user_fact(proceeding_type(PersonId, trial_charge)).
```

Listing 6.8 represents the transposition of Article 301 of the Polish Code of Criminal Procedure. This article states that the person who is suspected of having committed a crime is being interrogated by any authority, and has requested the presence of his/her defence counsel shall have the right to be assisted by a lawyer (rule r4). Rule r5 states that a person who has been informed of the charges against him but has not yet been formally charged in front of a court shall be granted the status of suspect.

Listing 6.9: User facts

```
% f1
user_fact(proceeding_matter(nino, interrogation)).
% f2
user_fact(proceeding_matter(nino, questioning)).
% f3
user_fact(person_request_submitted(nino, defence_counsel)).
% f4
user_fact(person_made_aware(nino, charge)).
% f5
user_fact(person_made_aware(nino, person_status)).
```

```
A0 : f4  ⟹  user_fact(person_made_aware(nino, charge))
A1 : f5  ⟹  user_fact(person_made_aware(nino, person_status))
A2 : f3  ⟹  user_fact(person_request_submitted(nino, defence_counsel)
     )
A3 : f1  ⟹  user_fact(proceeding_matter(nino, interrogation))
A4 : f2  ⟹  user_fact(proceeding_matter(nino, questioning))
A5 : A0,r3  ⟹  person_status(nino, accused)
A6 : A0,r5  ⟹  person_status(nino, suspect)
A7 : A5,A1,r1  ⟹  has_right(article4_a_iii, nino,
     right_to_information, privacy)
A8 : A5,A4,r2  ⟹  has_right(article6_3_a, nino,
     right_to_access_lawyer, questioning)
A9 : A6,A3,A2,r4  ⟹  has_right(art301, nino, right_to_access_lawyer,
     interrogation)
```

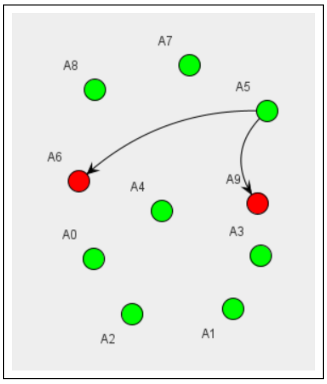


Figure 6.8: Base example

Listing 6.9 illustrates the facts of the case at hand. A person, named *nino*, has been called for interrogation (`f1`) and questioning (`f2`) by the legal authority, and has requested the presence of the defence counsel (`f3`). Furthermore, he has been made aware of his legal status (`f5`), and of the charges laid against him (`f4`).

Incorrect Transposition: diverging implementation

Listing 6.10: Rebuttal function integration

```
conflict([person_status(PersonId, suspect)], [person_status(
    PersonId, accused)]).
conflict([person_status(PersonId, accused)], [person_status(
    PersonId, suspect)]).
```

Listing 6.10 illustrates the conflicting relationship that exists between the two definitions of suspect and accused. It states that a person can be either a suspect or an accused, but not both at the same time. The user defined conflict can then be used while determining attacks between arguments. It is also important to notice how, for the purpose of this example, we exploited a feature of Arg2P allowing the use standard Prolog rules instead of the ASPIC-based ones. In the specific, rules in the Polish national law have been considered as defeasible in order to give priority to the EU rules and have the conflict resolved in favour of the latter.

Figure 6.8 shows the generated arguments on the left, and the visualisation of the results of the framework evaluation according to grounded semantic, on the right, on the basis of the facts added as input. Four arguments are of particular interest to the user: $A5$, $A6$, $A8$ and $A9$. $A5$ describes the EU definition of accused, while $A6$ the Polish definition of suspect. $A8$ and $A9$ represent the right of being assisted by a lawyer, which the system shows as being recognised by both legal sources. The presence of the conflict between the two definitions of suspect and accused applies, and it is represented as an attack-defeat relation between the two arguments $A5$ and $A6$, with a further attack on $A9$, which is the resulting right from the argument $A6$.

As the person is simultaneously an accused, according to the Directive, and a suspect, according to the Polish law, the two inferred conclusions are in direct conflict with each other, as illustrated by the arrows in the graph, but the European definition defeats the Polish one, due to the defeasible nature of the latter[4].

The result is not limited to the applicability of a certain rule, but can also help in understanding the source of the differences (and the conflicts) between the two legal sources. In our case, the user would be able to verify that the conflict between EU and Polish law derives from a different definition of the terms accused and suspect.

The user would furthermore realise that both arguments $A5$ and $A6$ are based on the same argument $A0$, the fact that the person has been made aware of the charges laid against him. We should also note the presence of argument $A7$, the right to be informed of his right to privacy, which will come into play in the next example.

Incorrect Transposition: conformity check

Listing 6.11: Conformity between directives and national laws check

```
generate :
    module(Module),
    prolog(call_module([Module, 'facts'],
```

[4] If both the Polish law and the Directive were to be transposed as defeasible norms, no argument would defeat the other, and the resulting conflict would be unresolved. If both were to be transposed as strict, no conflict can arise.

```
         with_facts_and_length(has_right(X, Y, Z, U), F, L))
            )
            => right(Module, X, Y, Z, U, F, L).

c0 : right(directive, X, PersonId, Right, U, F, L),
    ~(right(polish, XX, PersonId, Right, UU, FF, LL))
        => -conformity(polish, PersonId, Right).

c1 : right(polish, XX, PersonId, Right, UU, FF, LL),
    ~(right(directive, X, PersonId, Right, U, F, L))
        => -conformity(polish, PersonId, Right).

c2 : right(directive, X, PersonId, Right, U, F, L),
    right(polish, XX, PersonId, Right, UU, FF, LL)
        => conformity(polish, PersonId, Right).

module1 :-> module('directive').
module2 :-> module('polish').

conflict([right(directive, XX, A, Z, U, F, L)], [right(
   polish, X, A, Z, UU, FF, LL)]) :-
    \+ conflictFunction(F, FF).
conflict([right(polish, XX, A, Z, U, F, L)], [right(
   directive, X, A, Z, UU, FF, LL)]) :-
    \+ conflictFunction(FF, F).

conflictFunction(F, FF) :-
    sameFacts(FF, F),
    sameFacts(F, FF).

sameFacts([], _).
sameFacts([H|T], Facts) :-
    member(H, Facts),
    sameFacts(T, Facts).
```

```
A0 : module1  ⟹  module(directive)
A1 : module2  ⟹  module(polish)
A2 : A0,generate  ⟹  right(directive, article4_a_iii, nino,
     right_to_information, privacy, [person_made_aware(nino, person_status),
     person_made_aware(nino, charge)], 2)
A3 : A0,generate  ⟹  right(directive, article6_3_a, nino,
     right_to_access_lawyer, questioning, [proceeding_matter(nino, questioning
     ), person_made_aware(nino, charge)], 2)
A4 : A1,generate  ⟹  right(polish, art301, nino, right_to_access_lawyer,
     interrogation, [person_request_submitted(nino, defence_counsel),
     proceeding_matter(nino, interrogation), person_made_aware(nino, charge)],
      3)
A5 : A3,c0  ⟹  -conformity(polish, nino, right_to_access_lawyer)
A6 : A4,c1  ⟹  -conformity(polish, nino, right_to_access_lawyer)
A7 : A2,c0  ⟹  -conformity(polish, nino, right_to_information)
A8 : A3,A4,c2  ⟹  conformity(polish, nino, right_to_access_lawyer)
```

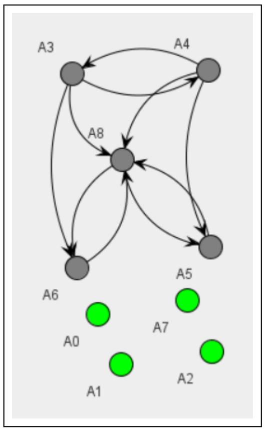


Figure 6.9: Conformity example

We shall now illustrate the use of argumentation to verify the conformity between directives and national laws (Listing 6.11).

The conflict, here, resides in the set of facts exploited by the directive and by the Polish national law to obtain a right. The system shall verify whether the facts in the two sets are equal. The variables `F` and `L` stand for facts and length respectively and are populated by the facts that are used by the system for reaching the goal.

We have also added three new rules: `c0` and `c1`, that verify that the national law conforms with the Directive; and `c2`, that verifies the lack of such conformity. In the example, these rules shall check whether the Polish national law conforms with the Directive.

Note that in this case the Prolog-like syntax evaluation has not been used. Instead, the Prolog code in the modules is evaluated through the rule `generate`. The evaluation of the Prolog goal `has_right` is done thanks to the special Arg2P predicate `prolog/1` allowing the evaluation of pure Prolog code inside the ASPIC-like syntax. The results obtained from the deductive reasoning are then exploited to build the arguments for the predicates `right_[name of the right]`. Argumentation enhances the level of explainability of the system. The rules to be applied in this case, and the facts that are needed as input, are the same as in Figure 6.8, with the difference that no source of law shall have priority over the other, thus all rules are written using the strict syntax.

Figure 6.9 is used to showcase, first of all, with regard to the right to be informed of his right to privacy, the most basic result that the system can give when evaluating the conformity of a national source of law with the European Directive. According to argument $A2$ the right exists following Article 4(a_iii), and the only inference based on that argument is found in $A7$, which illustrates the missing conformity of the Polish law with the Directive, as no similar right was returned from the Polish law. Both arguments are green as there are no attacks on any of its inferences.

Figure 6.9 also showcases the conformity evaluation of the system with regard to the right to be assisted by a lawyer. Both the Directive ($A3$) and the Polish law ($A4$) have returned the same right, although different conditions are needed to reach that result.

The facts which the system takes into account are only those manually added by the user. Thus, using the conformity evaluation the system does not take into consideration the different definitions of suspect and accused by the EU and Polish legislators, but bases its result on the `user_facts`. The results presented by the system show that the conditions required by the Directive include that the proceeding shall be the questioning of the defendant, while the Polish legislator refers to any interrogation, adding the requirement that the suspect shall explicitly request the presence of the defence counsel. The fact that the person has been made aware of the charges laid against him, which in Figure 6.8 was the source of the conflict between the two legal sources, in this example is the only condition they have in common, as the different definition of the status of the defendant is not relevant for the purpose of our conformity evaluation.

To summarise, on the one hand the system finds that the same right is returned by both the European and Polish modules, on the other hand, neither implementation contains the same requirements for the right to be guaranteed, therefore the system cannot decide whether the Polish law has successfully implemented the Directive.

The graph shows that the European right ($A3$) is in conflict with the Polish right ($A4$), and that both arguments attack the argument for conformity ($A8$), as the latter is not a perfect transposition of the former. Both Polish rights also attack the two arguments for the `-conformity`, as they are indeed returning the same right as the Directive. The system thus cannot conclude in one way or the other, and the arguments remain greyed out.

The user would therefore be provided with the information that, although the right exists in the Polish legal system, we cannot say whether the right has been fully implemented in the national legal system, thus an issue of applicability shall arise. Highlighting such contrasts can provide a better understanding of any underlining legal concerns that a traditional expert system cannot easily provide.

This feature could be particularly helpful, from a comparative perspective, in reaching a more uniform interpretation of the European legal source. With regard to Directives, as that is the object of this discussion, we can take into consideration the greater impact the judge may have on the proceedings in case of a negative transposition. Whenever an interpretation issue arises, and is made clear to the judiciary, the latter can intervene and look more in depth at the source in order to verify its applicability, as European legislative acts may be either directly applicable, thus binding in Member States, or they may be subject to an express implementation act of the national legislator.

Chapter 7

Beyond symbolic AI: Argumentation for ML

In our previous chapters, we focused on formal argumentation, introducing a new model and algorithms along with a new technology for argumentation. These developments have been shown to be effective in modeling and reasoning with legal concepts. However, there is an important shift occurring in the field of AI. The traditional, symbolic methods we have been discussing are no longer at the forefront. Instead, sub-symbolic methods, primarily those based on machine learning (ML), are emerging as key players. These methods excel in areas where symbolic approaches have historically struggled.

Yet, these ML methods come with their own set of challenges, particularly in sensitive domains like law. Their complexity can make them opaque and difficult to trust, for both specialists and general users. In this context, our traditional, logic-based symbolic methods retain their relevance, offering clarity and understanding in these complex systems.

Indeed, in legal settings, the application of machine learning (ML) systems, particularly in areas such as predictive justice, highlights the need for stringent ethical standards and clear compliance with legal norms. As we delve deeper into this context, several key aspects emerge. First, in the legal domain, ML systems must adhere to ethical guidelines that prevent biases and ensure fairness. This is critical because ML algorithms can inadvertently perpetuate or even amplify existing biases present in the data they are trained on. Ethical constraints involve implementing measures to detect and mitigate such biases, ensuring that the ML system's outcomes do not discriminate against any group or individual. Also, it is essential to have a transparent record of the principles and objectives that guided the development of an ML system. This clarity is crucial for understanding the system's intended purpose and the rationale behind specific design and implementation choices. In the legal field, where the reasons behind a decision can be as important as the decision itself, understanding the guiding principles of an ML system is indispensable. Moreover, transparency throughout the ML development process helps in identifying and addressing potential issues at various stages—from data collection and processing to model training and validation. This transparency is not just about making the system's mechanisms understandable but also about documenting the process, including how data is sourced, how models are selected and trained, and how decisions are made. While the final ML system might still retain a degree of opacity in its operations, having control over the process – especially from a data and behaviour standpoint – is crucial. This control involves understanding the data that feeds into the system, how the system processes this data, and how it arrives at its conclusions. It is about ensuring that the system behaves as expected under various circumstances and that any deviations or errors can be traced and understood.

Linking this to the European Union's AI Act, these aspects align closely with the proposed regulations. The AI Act aims to ensure that AI systems are safe and respect existing laws on fundamental rights and values. It emphasises transparency, accountability, and oversight of high-risk AI systems, including those used in legal and judicial contexts. The Act requires high-risk AI systems to undergo rigorous testing and certification processes, ensuring that they meet specific standards of accuracy and robustness.

Our final chapter explores how argumentation can be applied to the development of ML systems, but with a different angle from what's typically seen in Explainable AI (XAI). Instead of making AI systems more understandable to users, we aim to use argumentation to guide the development process itself with the goal of addressing the above mentioned challenges. We introduce automated machine learning (AutoML) as a key concept – a higher-level approach to machine learning that seeks to automate the process of selecting and tuning ML models. While AutoML simplifies some aspects of ML development, it can also lead to reduced control and understanding for data scientists, adding another layer of opacity. In the remaining of this chapter, we will discuss how structured argumentation can help address these challenges, offering Data Scientists a symbolic interface to input their expertise and steer the ML system's development in a transparent manner.

Let us start refreshing some notions already introduced in Section 2.3. In a machine learning (ML) project, the Data Scientist (DS) starts by gathering raw data from various sources, like data lakes. They gain an understanding of both the data and the problem at hand, and convert this understanding into specific constraints. This leads to the design and training of an ML model, which is eventually integrated into the existing data platform. This integration forms an ML pipeline, which includes a series of data pre-processing steps followed by the ML task. The DS selects from a wide range of algorithms and sets numerous hyperparameters. The effectiveness of the solution hinges on choosing the optimal combination of algorithms and hyperparameters from a vast array of possibilities.

Automated machine learning (AutoML) tools aid the DS in constructing the ML pipeline. These tools use advanced optimization techniques to efficiently navigate the extensive solution space. AutoML is known for delivering accurate results even with limited time or iterations. It's crucial for the DS to input their domain knowledge into the system to guide the AutoML tool away from invalid solutions. Despite this, the complexity of AutoML tools can be overwhelming, making it challenging for the DS to fully grasp and control the process, as noted in the work by Xin et al. [Xin et al., 2021] on AutoML automation challenges.

The need for a Human-centered and explainable framework for AutoML is real [Gil et al., 2019, Lee and Macke, 2020, Wang et al., 2019a] (or even mandatory in recent analytic scenarios where the user is interacting with mixed-reality and smart assistants [Francia et al., 2019, Francia et al., 2022]). It is crucial for the DS to augment her knowledge by learning new insights (e.g., new constraints) from the retrieved solutions. Indeed, the DS requires understanding the AutoML process in order to trust the proposed solutions [Drozdal et al., 2020]. Some works [Gil et al., 2019, Lee and Macke, 2020, Wang et al., 2019a] prescribe the usage of a Human-centered framework for AutoML, yet they only suggest design require-

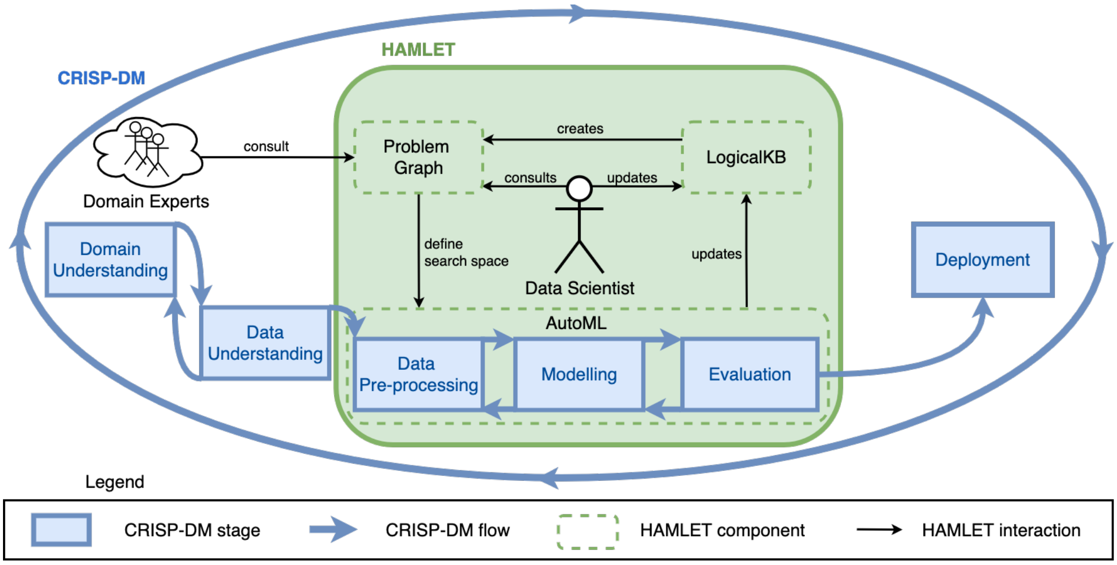


Figure 7.1: Integrating HAMLET with the CRISP-DM process model.

ments. Alternatively, the authors in [Ono et al., 2021] have proposed a tool that visualizes the best and the worst solutions retrieved by an AutoML tool. We claim that a Human-centered framework should provide the mechanisms to: (i) help the DS to structure her knowledge about the problem in an effective search space; and (ii) augment the knowledge initially possessed by the DS with the one produced by the AutoML optimization process.

For this purpose, we introduce HAMLET (Human-centered AutoMl via Logic and argumEnTation; Figure 7.1), a framework that enhances AutoML with structured argumentation to: structure the constraints and the AutoML solutions in a Logical Knowledge Base (LogicalKB); parse the LogicalKB into a human- and machine-readable medium called Problem Graph; devise the AutoML search space from the Problem Graph; and leverage the Problem Graph to allow both the DS and an AutoML tool to revise the current knowledge.

7.1 Problem Formulation

HAMLET intersects two research areas, *automated machine learning* and *argumentation*. Once defined the right language for encoding the DS and AutoML knowledge, a structured argumentation model (e.g., an ASPIC$^+$ instance

[Calegari et al., 2021c]) can support HAMLET with the formal machinery to build an argumentation framework upon the data, while abstract argumentation would dispense the evaluation tools.

Figure 7.1 illustrates the overview of HAMLET. When addressing end-to-end data analysis, a DS usually follows a process model such as CRISP-DM. The DS starts by collecting raw data in an arbitrary format. Then, "Domain Understanding" is conducted. The DS works in close cooperation with domain experts and enlists *domain-related constraints* (i.e., intrinsic of the problem). Follows "Data Understanding", devoted to data analysis, and to extract *data-related constraints* (e.g., defined by the data format). Domain and Data Understanding might be repeated many times until the DS is satisfied by the acquired knowledge. Once confident, the DS investigates different solutions throughout "Data Pre-processing", "Modelling", and "Evaluation". Data Pre-processing and Modelling are conducted to effectively build the solution, while Evaluation offers a way to measure its performance. Such a solution consists of a *ML pipeline*: a sequence of *Data Pre-processing transformations* ending with an *ML task*. The DS instantiates different pipelines among a large set of algorithms; the performance are affected by both the algorithms and some exposed hyperparameters. While seeking the best performing and valid solution, the DS should consider the already known constraints – domain- and data-related – and the ones she discovers during Data Pre-processing and Modelling, respectively: *transformation-* and *algorithm-related constraints* (e.g., due to the intrinsic semantic of transformations and algorithms at hand). Finally, the process concludes with the "Deployment" of the solution.

HAMLET intersects CRISP-DM, allowing the DS to inject and augment her knowledge while automatizing the exploration towards the solution (i.e., instantiate the best ML pipeline). We now dig the foundation of HAMLET by incrementally introducing the concepts necessary to move from AutoML to logic and argumentation. To support the reader, we summarize the main notation in Table 7.1.

We provide a formalization necessary to move from single algorithms to the optimal pipeline. For the sake of clarity, we refer to a Classification task, but the formalization also holds for supervised ML tasks in general.

Table 7.1: Main symbols used in the formalization.

Symbol	**Meaning**
A	Algorithm
h	Algorithm hyperparameter
S	Step
P	Pipeline
λ_*	Instance of *
Λ_*	Domain of *
Λ	Search space

Definition 49 (Dataset). *A* dataset X *is a matrix where data items (i.e., rows) are characterized by features (i.e., columns).*

Definition 50 (Algorithm). *An* algorithm A *is a function that transforms an input dataset* X' *into a new dataset* X''*. The algorithm exposes a (possibly empty) set* H *of* hyperparameters*. Each hyperparameter* $h \in H$ *has a* domain Λ_h*. We call the* algorithm domain Λ_A *the Cartesian product of all hyperparameter domains (i.e.,* $\Lambda_A = \Lambda_{h_1} \times \ldots \times \Lambda_{h_{|H|}}$*). We call* algorithm instance $\lambda_A \in \Lambda_A$ *an algorithm whose hyperparameters have been assigned with values from their respective domains.*

A Classification algorithm returns a vector (i.e., a matrix with a single column) of labels Y out of the input dataset X'.

Definition 51 (Step). *A* step S *is a set of alternative algorithms with the same goal. The* step domain *is defined as a disjoint union of the algorithm domains* $\Lambda_S = \Lambda_{A_1} \uplus \ldots \uplus \Lambda_{A_{|S|}}$.

Where $\uplus$ combines the domains of the given algorithms, while retaining the original domain membership (i.e., it is possible to refer to the domain of each algorithm included in a step).

We identify two types of steps: Data Pre-preprocessing steps (e.g., Discretization, Normalization) shape the dataset for the last mandatory step, which fulfill the task—Classification in this case.

Example 16 (Algorithm and step). *Examples of steps are Normalization ($\mathcal{N}$), Discretization ($\mathcal{D}$), and Classification ($\mathcal{C}l$). An algorithm for Classification is Decision Tree ($\mathcal{D}t$) [Breiman et al., 1984], examples of hyperparameters for $\mathcal{D}t$ are* *its maximum* depth ($\mathbb{N}^+$) *and the minimum* samples split ($\mathbb{N}^+$) *required to split a node; hence* $\Lambda_{\mathcal{D}t} = \mathbb{N}^+ \times \mathbb{N}^+$. *An example of algorithm instance is* $\lambda_{\mathcal{D}t} = \{\text{depth} = 3, \text{samples\_split} = 10\}$.

Definition 52 (Pipeline). *Given a (possibly empty) set of Pre-processing steps* $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$ *and a Classification algorithm* A *from the Classification step, a* pipeline P *is a sequence that concatenates steps from* $\mathcal{S}$ *and* A. *The domain of a pipeline is* $\Lambda_P = \Lambda_{S_1} \times \ldots \times \Lambda_{S_{|\mathcal{S}|}} \times \Lambda_A$. *We call* pipeline instance λ_P *a sequence of algorithm instances* $\lambda_P = \langle \lambda_{A_1}, \ldots, \lambda_{A_{|P|}} \rangle$ *such that* $\lambda_P \in \Lambda_P$.

Example 17 (Pipeline and pipeline instance). *Given the pre-processing steps Normalization ($\mathcal{N}$) and Discretization ($\mathcal{D}$), the possible pipelines for the DecisionTree ($\mathcal{D}t$) are:*

$$
\begin{aligned}
&P_1 = \langle \mathcal{D}t \rangle \qquad && P_2 = \langle \mathcal{D}, \mathcal{D}t \rangle \qquad && P_4 = \langle \mathcal{D}, \mathcal{N}, \mathcal{D}t \rangle \\
& && P_3 = \langle \mathcal{N}, \mathcal{D}t \rangle && P_5 = \langle \mathcal{N}, \mathcal{D}, \mathcal{D}t \rangle
\end{aligned}
$$

Given Binarizer *($\mathcal{B}$) and* KBinsDiscretizer *($\mathcal{K}b$) as algorithms of Discretization ($\mathcal{D}$), and* MinMaxScaler *($\mathcal{M}m$) and* StandardScaler *($\mathcal{S}s$) and as algorithms of Normalization ($\mathcal{N}$), we provide examples of instances of P_2 and P_4:*

$$
\begin{aligned}
\lambda_{P_2} &= \langle \lambda_{\mathcal{B}},\ \lambda_{\mathcal{D}t} \rangle, & \lambda_{P_4} &= \langle \lambda_{\mathcal{K}b},\ \lambda_{\mathcal{M}m},\ \lambda_{\mathcal{D}t} \rangle \\
\lambda_{\mathcal{B}} &= \{\text{thr} = 5.5\}, & \lambda_{\mathcal{K}b} &= \{\text{n\_bins} = 3, \ldots\} \\
\lambda_{\mathcal{M}m} &= \{\varnothing\}, & \lambda_{\mathcal{D}t} &= \{\text{depth} = 3, \ldots\}
\end{aligned}
$$

Figure 7.2 depicts the pipeline domain Λ_{P_4} *and the pipeline instance* λ_{P_4}.

Depending the on the involved algorithms, their order and hyperparameters, the search space – out of which the best pipeline instance is select – is defined as follows. While, its extraction is later discussed in Algorithm 1.

Definition 53 (Search space). *The* search space Λ *is the Cartesian product of the domain of the Classification step and the disjoint union of the all partial permutations of the pre-preprocessing steps domains.*

AutoML optimizes the exploration of such space. However, it is not only about algorithms and hyperparameters but also about constraints.

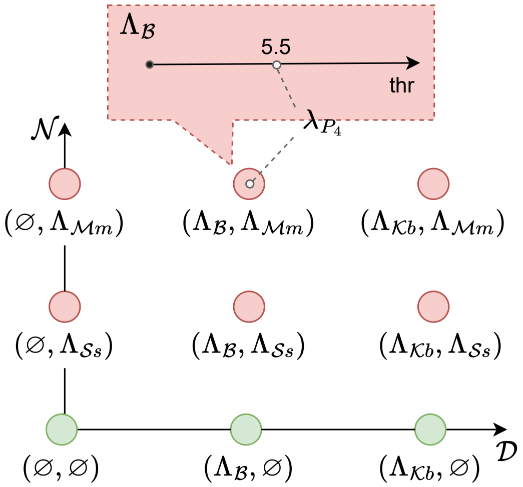

$\Lambda_{\mathcal{B}}$
5.5
thr
λ_{P_4}
$\mathcal{N}$
$(\varnothing, \Lambda_{\mathcal{M}m})$
$(\Lambda_{\mathcal{B}}, \Lambda_{\mathcal{M}m})$
$(\Lambda_{\mathcal{K}b}, \Lambda_{\mathcal{M}m})$
$(\varnothing, \Lambda_{\mathcal{S}s})$
$(\Lambda_{\mathcal{B}}, \Lambda_{\mathcal{S}s})$
$(\Lambda_{\mathcal{K}b}, \Lambda_{\mathcal{S}s})$
$(\varnothing, \varnothing)$
$(\Lambda_{\mathcal{B}}, \varnothing)$
$(\Lambda_{\mathcal{K}b}, \varnothing)$
$\mathcal{D}$

Figure 7.2: Examples of the pipeline domain Λ_{P_4} and pipeline instance λ_{P_4}, for the sake of visualization we omit the third dimension representing the domain of the Decision Tree. Green (or red) circles represent valid (or invalid) sub-regions of the search space; Normalization is not allowed in the pipeline. The rectangle represents a zoom in the domain of the Binarizer algorithm.

Definition 54 (Constraint)**.** *A* constraint $C \subseteq \Lambda$ *is a region of search space that is either* mandatory *or* forbidden*. Given a pipeline instance* $\lambda_P \in \Lambda_P \subseteq \Lambda$

- *a mandatory constraint C is fulfilled if $\lambda_P \in C$;*
- *a forbidden constraint C is fulfilled if $\lambda_P \notin C$.*

Example 18 (Constraint)**.** *Given the Normalization step ($\mathcal{N}$) and Decision Tree ($\mathcal{D}t$) as a Classification algorithm, an example of algorithm-related constraint is "forbid $\mathcal{N}$ in pipelines with $\mathcal{D}t$". This discards all the pipelines containing both Normalization and Decision Tree. Figure 7.2 depicts the effects of the constraint on the pipeline domain Λ_{P_4}.*

Considering all the constraint combinations is overwhelming and, additionally, *conflicts* might occur; for instance in the case of ethical [Harrison and Rubinfeld, 1978] and legal fields that easily inject conflicting constraints into the search space.

Definition 55 (Constrained pipelines optimization)**.** *Given a search space* Λ *and a set of constraints* $\mathcal{C}$*, finding the best pipeline instance* $\hat{\lambda}_P$ *is defined as* $\hat{\lambda}_P =$

$argmax_{\lambda_P \in \Lambda_P} metric(\lambda_P)$, where $metric(\lambda_P)$ is the function evaluating the goodness of λ_P and the explored pipelines fulfill the constraints in $\mathcal{C}$.

AutoML is not explainable, hence it does not provide the DS with feedbacks that would help her to augment the knowledge about the problem. It is necessary to represent both (i) the DS knowledge about the problem and (ii) the outcome of the AutoML tool in a uniform human-readable medium. The former helps to drive the optimization process, the later augments the knowledge about the problem by learning from the explored configurations of pipeline instances—deriving new constraints that increase the DS awareness.

We leverage argumentation as the key element in defining a common structure (i.e., a uniformed human- and machine-readable medium) on which the knowledge of both the DS and the AutoML tool can be combined fruitfully. In a way, our approach follows the steps of the well known logical based expert systems, of which it is possible to find a great number of successful examples [Tan, 2017]. Argumentation provides the tools to cope with one of the distinctive features of the knowledge we want to deal with: inconsistency. Indeed, the ML process is the product of possible attempts, validated or refuted by a consequent evaluation. Hence, the mechanism used to encode the knowledge is required to manage this constant revision process.

Let us start from the Argumentation Framework described in Section 2.1. Remembering the concept of an Argumentation System as outlined in Definition 3, we need to establish a Logical Language L, which will form the foundation for the set of rules R used to define how elements from the language are combined together, and a conflict function $\rhd$. In the following two definitions, we specialize L into the language L_{ML} expressing all the basic elements of an AutoML problem and R into a Logical Knowledge Base written in the language L_{ML}.

Definition 56 (AutoML language). *Given an argumentation language L, we define the* AutoML language *L_{ML} as $L \cup W$, with W the following set of predicates*[1]*:*

- *step(S) with $S \in L$, representing a step S in the pipeline;*

[1] For the sake of conciseness, when writing statements of the AutoML language, the letters S (and A) refer to the name of the step (and algorithm)

- *algorithm(S, A) with $S, A \in L$, representing an algorithm A for the step S;*

- *hyperparameter(A, h, t) with $A, h, t \in L$, representing an hyperparameter h for the algorithm A of type t (e.g., numerical, categorical);*

- *domain(A, h, Λ_h) with $A, h, \Lambda_h \in L$, representing an hyperparameter h for the algorithm A with domain Λ_h;*

- *pipeline($\langle S_1, \ldots, S_n \rangle$, A) with $S_1, \ldots, S_n, A \in L$, representing a pipeline consisting of the sequence of steps $\langle S_1, \ldots, S_n \rangle$ and the Classification algorithm A;*

- *mandatory($\langle S_1, \ldots, S_n \rangle$, Z) with $S_1, \ldots, S_n, Z \in L$, representing a constraint imposing the steps $\langle S_1, \ldots, S_n \rangle$ on the pipelines with algorithm A ($Z = A$) or on all the Classification pipelines ($Z = \mathcal{C}l$);*

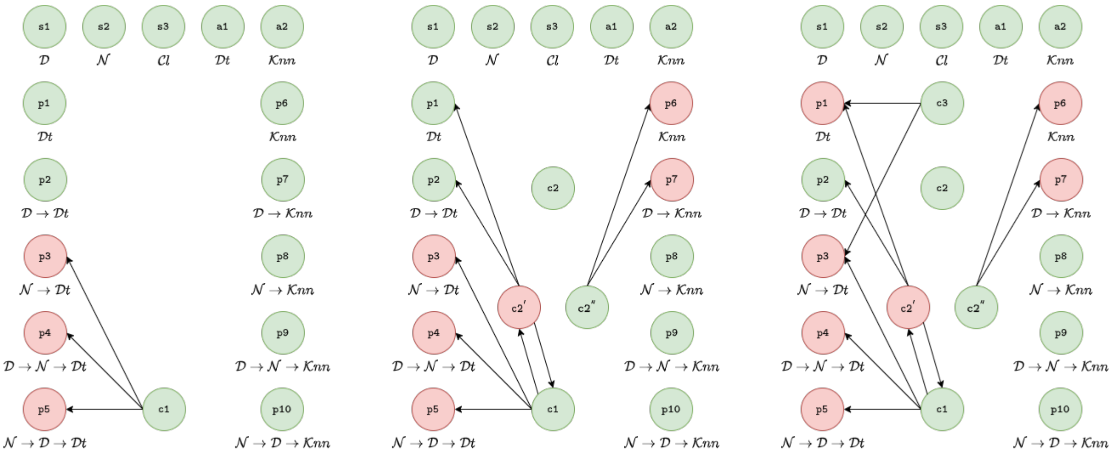
- *forbidden($\langle S_1, \ldots, S_n \rangle$, Z) with $S_1, \ldots, S_n, Z \in L$, representing a constraint forbidding the steps $\langle S_1, \ldots, S_n \rangle$ on the pipelines with algorithm A ($Z = A$) or on all the Classification pipelines ($Z = \mathcal{C}l$);*

- *mandatory_order($\langle S_1, \ldots, S_n \rangle$, Z) with $S_1, \ldots, S_n, Z \in L$, representing a constraint imposing the sequence of steps $\langle S_1, \ldots, S_n \rangle$ on the pipelines with algorithm A ($Z = A$) or on all the Classification pipelines ($Z = \mathcal{C}l$).*

Definition 57 (Logical Knowledge Base)**.** *Given the language L_{ML}, we call Logical Knowledge Base (LogicalKB) the set of rules for a given AutoML problem.*

In other words, the DS leverages an intuitive logical language (i.e., L_{ML}), and enlists the constraints one-by-one (i.e., in the LogicalKB). In our vision, the LogicalKB consists of (i) a set rules specified by the DS and a (ii) set of common rules that enable the automatic derivation of pipelines and constraints. Besides, the DS community could create a shared LogicalKB derived from the available literature and similar real-case problems.

Example 19 (Logical Knowledge Base)**.** *We focus on Discretization ($\mathcal{D}$), Normalization ($\mathcal{N}$) and Classification ($\mathcal{C}l$) steps, and, for brevity, only define the Classification algorithms: Decision Tree ($\mathcal{D}t$) and K-Nearest Neighbors ($\mathcal{K}nn$).*



(a) (b) (c)

Figure 7.3: Examples of Problem Graphs. Green nodes are valid arguments, red ones are refuted. Arrows are attacks.

```
# define Discretization step
s1 : ⇒ step(𝒟).
# define Normalization step
s2 : ⇒ step(𝒩).
# define Classification step
s3 : ⇒ step(𝒞l).
# DT is a Classification algorithm
a1 : ⇒ algorithm(𝒞l, 𝒟t).
# Knn is a Classification algorithm
a2 : ⇒ algorithm(𝒞l, 𝒦nn).
# Forbid Normalization when using DT
c1 : ⇒ forbidden(⟨𝒩⟩, 𝒟t).
```

s1, s2, and s3 represent the steps; a1 and a2 represent the algorithms; finally, c1 represent the algorithm-related constraint from Example 18, namely “forbid $\mathcal{N}$ in pipelines with $\mathcal{D}t$”.

When applying constraints, they can be conflicting. The conflict relation $\rhd$ reifies the constraints from Definition 54 through the conflict function c_{ML}.

Definition 58 (AutoML Conflict)**.** *The conflict function c_{ML} is a function from*

L_{ML} to $2^{L_{ML}}$ that given a statement from L_{ML} returns the set of conflicting statements.

We support both the AutoML conflicts on "pipeline vs constraint" and "constraint vs constraint". Formally, let us consider two lists of steps $\alpha = \langle \ldots, S_i, S_j, \ldots \rangle$ and $\beta = \langle \ldots, S_y, S_x, \ldots \rangle$.

- Pipeline vs constraint: return the constraints conflicting with pipelines.

$$\begin{aligned} & c_{ML}(\mathit{pipeline}(\beta, A)) = \\ & \quad \{\mathit{mandatory}(\alpha, A) \mid \exists S_i \in \alpha \ s.t. \ S_i \notin \beta\} \cup \\ & \quad \{\mathit{forbidden}(\alpha, A) \mid \forall S_i \in \alpha, \ S_i \in \beta\} \cup \\ & \quad \{\mathit{mandatory\_order}(\alpha, A) \mid \exists S_i, S_j \in \alpha, S_x, S_y \in \beta, \\ & \qquad\qquad S_i = S_x, S_j = S_y \ s.t. \ i < j, x > y\} \end{aligned}$$

  Intuitively, a pipeline *pipeline*$(\langle S_i, S_j \rangle, A)$ is conflicting with a *mandatory* constraint if the pipeline does not contains at least a mandatory step (e.g., the pipeline is conflicting with *mandatory*$(\langle S_j, S_k \rangle, A)$), with a $forbidden$ constraint if the pipeline contains all the forbidden steps (e.g., the pipeline is conflicting with $forbidden(\langle S_j \rangle, A)$), and with a *mandatory_order* constraint if the pipeline contains at least two steps that are not in the mandatory order (e.g., the pipeline is conflicting with *mandatory_order*$(\langle S_j, S_i \rangle, A)$).

- Constraint vs constraint: return the constraints conflicting with other constraints.

$$\begin{aligned} & c_{ML}(\mathit{forbidden}(\beta, A)) = \{\mathit{mandatory}(\alpha, A) \mid \forall S_j \in \beta, S_j \in \alpha\} \\ & c_{ML}(\mathit{mandatory}(\beta, A)) = \{\mathit{forbidden}(\alpha, A) \mid \forall S_j \in \alpha, S_j \in \beta\} \\ & c_{ML}(\mathit{mandatory\_order}(\beta, A)) = \\ & \quad \{\mathit{mandatory\_order}(\alpha, A) \mid \exists S_i, S_j \in \alpha, S_x, S_y \in \beta, \\ & \qquad\qquad S_i = S_x, S_j = S_y \ s.t. \ i < j, x > y\} \end{aligned}$$

  Intuitively, *mandatory* and $forbidden$ constraints are in conflict if all the forbidden steps are included in the mandatory constraint (i.e., *mandatory*$(\langle S_i, S_j, S_k \rangle, A))$

and a $forbidden(\langle S_i, S_j \rangle, A)))$, this hold symmetrically for $forbidden$ and $mandatory$ constraints. Two $mandatory\_order$ constraints are in conflict if they contain at least two steps in different order (i.e., $mandatory\_order(\langle S_i, S_j, S_k \rangle, A))$ and a $mandatory\_order(\langle S_j, S_i \rangle, A)))$.

Example 20 (AutoML conflict)**.** *With reference to the LogicalKB in Example 19, let us consider the set of rules that represent the pipelines related to* $\mathcal{D}t$*:*

```
# pipeline ending with a DT
p1 : ⇒ pipeline(𝒟t).
# Discretization and DT
p2 : ⇒ pipeline(⟨𝒟⟩, 𝒟t).
# Normalization and DT
p3 : ⇒ pipeline(⟨𝒩⟩, 𝒟t).
# Discretization, Normalization, and DT
p4 : ⇒ pipeline(⟨𝒟, 𝒩⟩, 𝒟t).
# Normalization, Discretization, and DT
p5 : ⇒ pipeline(⟨𝒩, 𝒟⟩, 𝒟t).
```

In this case, `c1` *(i.e., "forbid* $\mathcal{N}$ *in pipelines with* $\mathcal{D}t$*") is in conflict with the* pipeline *statements* `p3`*,* `p4`*, and* `p5` *since they contain* $\mathcal{N}$ *and* $\mathcal{D}t$*.*

Starting from the language L_{ML}, a set of rules in this language, and the conflict function c_{ML}, we can derive arguments and attacks – and consequently the entire argumentation framework – as per Definition 6, Definition 8 and Definition 9. The evaluation of the argumentation framework is performed through Dung's grounded semantics [Dung, 1995a].

Throughout the rest of the chapter we will refer to an argument with the set of rules used to generate it (e.g., given $r :\Rightarrow c$ and $r1 : c \Rightarrow d$, we will write r and $r1$ when referring to the rules and $\{r\}$ and $\{r, r1\}$ for – respectively – the argument with *conclusion* c using r and the argument with *conclusion* d using r and $r1$). We also refer to the graph in which nodes are arguments and edges are attacks from the argumentation framework as *Problem Graph*.

The benefits of the Problem Graph are two-fold. First of all, it can be leveraged by both DSs and domain experts to: understand, summarize and visualize the current knowledge. Second of all, it is straightforward to convert such a graph of constraints into a space of possible solutions (i.e., exploiting argumentation semantics, it is easy to obtain all the sets of arguments – constraints and pipelines – which hold together).

Example 21 (Problem Graph)**.** *Figure 7.3a illustrates the Problem Graph extracted from the LogicalKB introduced in Example 19 and 20 and evaluated under grounded semantics. Arguments are represented as nodes, attacks as arrows and the colors represent the state of the arguments according to the semantics: red for refuted arguments, and green for the ones in the extension. The arguments are identified through the set of rules used to build them. In the upper part of the figure, we have a group of undefeated arguments, namely {`s1`}, {`s2`}, {`s3`}, {`a1`}, and {`a2`}, representing the basic knowledge used to setup the AutoML search space (i.e. steps and algorithms). Then, we have an argument for every pipeline in Example 20: from {`p1`} to {`p5`} the pipelines regarding $\mathcal{D}t$, from {`p6`} to {`p10`} the ones regarding $\mathcal{K}nn$. Finally, we can observe three different attacks: from {`c1`} to {`p3`}, {`p4`}, and {`p5`}, in accordance with the conflicts identified in Example 20. The arguments in the extension give us all the information that we should use during the AutoML optimization process – i.e. we should discard all the pipelines refuted by the constraint argument (*{c1}*), and focus on the remaining part of the search space.*

The use of argumentation relieves the DS of the burden of manually considering all the effects of the possible constraints. It is important to notice that, although the increased degree of automation, the Problem Graph allows the DS and domain experts to correct, revise, and supervise the process. Accordingly, possible inconsistencies – due to diverging constraints – can be verified by the DS using her knowledge.

Any change in the LogicalKB translates into a change in the Problem Graph, allowing the DS and domain experts to visualize it and argue about it. The revision of the Problem Graph is the key element in the process of augmenting the knowledge: the DS and domain experts can consult each other and discuss how the new insights relate to their initial knowledge. Indeed, thanks to the nature of the Problem Graph, it would be extremely easy to identify new possible conflicts and supporting arguments. Furthermore, AutoML can update the Problem Graph

```
# given an algorithm, create a pipeline including only such
   algorithm
hc0 : algorithm($\mathcal{C}l$, A) ⇒ pipeline($\langle\ \rangle$, A).
# given some steps and an algorithm, create a pipeline
   including such steps and algorithm
hc1 : step($S_1$),...,step($S_n$), algorithm($\mathcal{C}l$, A) ⇒ pipeline(
   $\langle S_1,\ldots,S_n\rangle$, A).
# given constraints on the Pre-processing steps required for
    Classification...
# ... apply this constraints to all Classification
   algorithms
hc2 : mandatory($\langle S_1,\ldots,S_n\rangle$, $\mathcal{C}l$), algorithm($\mathcal{C}l$, A) ⇒ mandatory(
   $\langle S_1,\ldots,S_n\rangle$, A).
hc3 : forbidden($\langle S_1,\ldots,S_n\rangle$, $\mathcal{C}l$), algorithm($\mathcal{C}l$, A) ⇒ forbidden(
   $\langle S_1,\ldots,S_n\rangle$, A).
hc4 : mandatory_order($\langle S_1,\ldots,S_n\rangle$, $\mathcal{C}l$), algorithm($\mathcal{C}l$, A) ⇒
   mandatory_order($\langle S_1,\ldots,S_n\rangle$, A).
```

Figure 7.4: A subset of rules from the LogicalKB.

by extracting constraints from the performed exploration, and transposing them into the LogicalKB. For instance, the DS may not have considered that the dataset contains missing values. AutoML helps in identifying the new data-related constraint "require Imputation ($\mathcal{I}$) in all the pipelines" and adds it to the LogicalKB ($mandatory(\langle\mathcal{I}\rangle, \mathcal{C}l)$).

The described process is compliant with and augments the CRISP-DM process. The inferred/learned knowledge is automatically handled throughout iterations, supporting the DS in the whole analysis in a continuous revision of the constraints.

7.2 HAMLET

HAMLET iterates over three phases (Figure 7.1): (i) the generation of Problem Graph and search space out of the LogicalKB, (ii) the exploration of the search space in compliance with the specified constraints, and (iii) the augmentation of the LogicalKB through a rule recommendation.

The framework is available at `https://github.com/QueueInc/HAMLET`, and it is composed of two sub-modules. The first, written in Kotlin and running on the

JVM, exposes a graphical interface on which the DSs can compile and revise the LogicalKB. The module is also responsible for the generation and evaluation of the Problem Graph; it implements the structured argumentation functionalities as specified in Section 7.1 using Arg2P [Calegari et al., 2021c]. The second module, written in Python, is responsible for performing the AutoML optimization and the extraction of the new constraints from the explored space.

7.2.1 Generation of Problem Graph and Search Space

In Section 7.1, we defined the LogicalKB as the set of rules specified by the DS using her knowledge. The LogicalKB also includes a set of hard-encoded rules representing inferences necessary to characterize the AutoML problems. These rules are joined to the ones defined by the DS and used to build the Problem Graph (i.e., argumentation framework).

A subset of rules is shown in Figure 7.4. The first two ($hc0$ and $hc1$) define how to automatically derive a pipeline using algorithms and steps. The construction of pipelines can be completely automated and the DS should be dispensed from manually enumerating all the possible pipelines as in Example 20. In particular, the correct set of rules is built dynamically using the steps and algorithms provided by the DS, then they are used to derive all the arguments for the possible pipelines. The last three rules ($hc2, hc3$ and $hc4$) encode constraints – mandatory, forbidden, mandatory_order – on all the available algorithms with a single statement (e.g., $mandatory(\langle\mathcal{D}\rangle, \mathcal{C}l)$): it will be automatically used by the framework to derive the constraints for all the specific algorithms in the theory.

Example 22 (Hard-coded rules)**.** *With reference to Example 21 and Figure 7.4, we add rule* `c2` *for a new data-related constraint.*

```
# mandatory Norm. in Class. pipelines
c2 : ⇒ mandatory(⟨𝒩⟩, 𝒞l)
```

From the rule `c2`*, the hard-coded rules generate the two arguments* `c2' =` $\{$`c2, a1, hc2`$\}$ *(i.e.,* $mandatory(\langle\mathcal{N}\rangle, \mathcal{D}t)$*) and* `c2'' =` $\{$`c2, a2, hc2`$\}$ *(i.e.,* $mandatory(\langle\mathcal{N}\rangle, \mathcal{K}nn)$*) that are specific for the Classification algorithms in the LogicalKB.*

However, $\{$`c1`$\}$ *(i.e.,* $forbidden(\langle\mathcal{N}\rangle, \mathcal{D}t)$*; is in conflict with* `c2'`*. Depending on her experience, the DS decides to resolve the conflict by specifying an ordering* *over the rules in the LogicalKB—the framework is based on ASPIC$^+$ thus it fully supports the use of preferences. Let us assume the use of the* last-weakest *ordering as defined in Definition 21. Assuming that the DS prefers* `c1` *to* `hc2`*, the argument* {`c1`} *is preferred to* `c2'` *and the attack from the latter is not considered in the final graph. Figure 7.3b shows the updated graph. Firstly, we observe the support relation between* {`c2`} *and the generated constraints* `c2'` *and* `c2''`*. Since* {`c1`} *has no attackers, it is added to extension. Consequently,* `c2'` *is refuted and the the pipelines attacked by it are correctly reinstated.*

Given the Problem Graph (we recall that the Problem Graph contains *all* the generated pipelines – including their partial permutations), the search space can be extracted as in Algorithm 1. We iterate over all the generated pipelines in the Problem Graph and we recursively build their domain: the pipeline domain is the Cartesian product of the step domains, the step domain is the disjoint union of the algorithm domains (we leverage the disjoint union since each algorithm can be picked as an alternative to the others), the algorithm domain is the Cartesian product of its hyperparameters; the domain of a hyperparameter is given by definition. Finally, the search space is the disjoint union of all the alternative pipeline domains.

Noticeably, while the search space could be constrained during its construction (e.g., by simply adding an "if" condition to check the validity of each pipeline at Algorithm 1 line 10), current AutoML frameworks leverage optimization techniques that do not allow the explicit exclusion of regions from the search space. As a consequence, we need to produce the entire search space first.

7.2.2 Exploration of a Constrained Search Space

The Problem Graph is not only used to build the entire search space but it is also evaluated to understand which pipelines are invalid and which constraints are valid. Hence – through the Problem Graph – we enhance AutoML exploration by combining the following techniques.

(i) Invalid pipelines are used to discourage the exploration of such a portion of the search space (we recall that a pipeline has a domain – a region of the

Algorithm 1 Search Space from the Problem Graph

```
Require: PG(N, E): Nodes and Edges of a Problem Graph
Ensure: $\Lambda$: Search Space

procedure GetDomain($A$)
    $\Lambda_A \leftarrow \varnothing$
    for each $h \in A$ do                          ▷ For each hyperparameter in the algorithm...
        $\Lambda_A \leftarrow \Lambda_A \times \Lambda_h$          ▷ Compute Cartesian product of hyperpar. domains
    end for
    return $\Lambda_A$                                  ▷ Return the algorithm domain
end procedure

$\Lambda \leftarrow \varnothing$                                   ▷ Initialize the search space
for each $pipeline(\alpha, A) \in N$ do    ▷ For each argument that is a pipeline with $\alpha$ steps and alg. $A$...
    $\Lambda_P \leftarrow$ GetDomain(A)             ▷ Init. pipeline domain with algorithm domain
    for each $S \in \alpha$ do                              ▷ For each step in the pipeline...
        $\Lambda_S \leftarrow \varnothing$                                  ▷ Init. the step domain
        for each $A \in S$ do                          ▷ For each algorithm in the step...
            $\Lambda_S \leftarrow \Lambda_S \uplus$ GetDomain(A)                ▷ Add alg. to step domain
        end for
        $\Lambda_P \leftarrow \Lambda_P \times \Lambda_S$               ▷ Add step domain to pipeline domain
    end for
    $\Lambda \leftarrow \Lambda \uplus \Lambda_P$                    ▷ Add pipeline domain to the search space
end for
return $\Lambda$                                          ▷ Return the search space
```

search space – in which several pipeline instances are parametrized). First, we sample such regions of the search space, then we enforce a knowledge injection mechanism through warm-starting (i.e., the process of providing previous evaluations that help the model to converge faster). For instance, with reference to Example 22, we sample some pipeline instances from the pipelines that have been discarded (from {p3} to {p7}); then, we label such samples as invalid and provide them to the AutoML tool, helping the optimization algorithm to focus only on the valid portions of the space.

(ii) Valid constraints – expressed as conjunctions of Boolean clauses – are used to discard the invalid pipeline instances that still are encountered by the AutoML tool. Indeed, since the sampling from (i) is non-exhaustive, it can happen that small portions of invalid regions could still be explored.

Our AutoML implementation is based on FLAML [Wang et al., 2021a], which mixes Bayesian Optimization with CFO (Frugal Optimization for Cost-related Hyperparameters). In a standard Bayesian process, an increasingly accurate model is built on top of the previously explored pipeline instances to suggest the most promising ones among the remaining. The pipeline instances keep being explored, updating the model, until a budget in terms of either iterations or time is reached. With CFO, there is also an estimation of the evaluation time to consider the frugality of the suggested pipeline instances – hence favoring the ones requiring a smaller amount of time. Throughout the exploration, different solutions are tested, which contribute to augmenting the global knowledge about the problem.

7.2.3 Knowledge Augmentation through Rule Recommendation

New constraints are automatically mined out of the pipeline instances explored by AutoML and *recommended* in our logical language as rules. Then, the DS decides which rules are accepted and added to the LogicalKB.

At this stage, we leverage frequent pattern mining techniques to learn constraints in an unsupervised manner. Frequent pattern mining is the task of finding the most frequent and relevant patterns in large datasets (e.g., finding the products frequently bought together in the domain of market basket analysis); depending on the constraint type, we look for (sub)sets [Srikant and Agrawal, 1995] or (sub)sequences [Srikant and Agrawal, 1996] frequently recurring among the explored pipelines. Since a pipeline instance is a sequence of algorithms, the set of the explored pipeline instances can be directly mapped into a transactional dataset [Srikant and Agrawal, 1995] where each pipeline instance is a transaction and each step – inferred from the algorithm – is an item.

We recommend the same constraints we support at the argumentation level (i.e., $mandatory$, $forbidden$, $mandatory\_order$) so that AutoML can be as expressive as the DS. For mandatory and forbidden constraints we look for (sub)sets [Srikant and Agrawal, 1995] frequently recurring among the explored pipelines. Specifically, we split the explored pipeline instances by the applied Classification algorithm, set a minimum frequency (i.e., support) threshold to 50% (i.e., to be retrieved, a set/sequence must occur at least in 50% of the explored instances), and extract frequent maximal[2] itemsets. The recommendation depends on the constraint.

- *mandatory*: we consider only the patterns with good performance (i.e., $0.7 \leq metric \leq 1.0$);
- *forbidden*: we consider only the patterns with bad performance (i.e., $0.0 \leq metric \leq 0.3$);
- *mandatory_order*: the same considerations of the mandatory constraints stand, except that we look for (sub)sequences [Srikant and Agrawal, 1996] of length 2 to discover ordering dependencies in pairs of steps as in [Giovanelli et al., 2021a].

We leveraged well-known implementations [Raschka, 2018] and [Wang et al., 2022] for itemsets and sequences mining, respectively. Finally, we return to the DS only the top-10 rules sorted by descending support; we allow the DS to explore all the rules on-demand.

The thresholds act as filters on the extracted rules since we cannot burden the user with the investigation of hundreds of recommendations. As to the intervals, our rationale is simple: we only want to recommend as mandatory (order) the rules that achieved "good performance" and as forbidden the rules that achieved "bad performance". Since we handle classification pipelines that mainly refer to (balanced) accuracy/F1 score/recall, we mapped "good" in the interval $[0.7, 1.0]$ and "bad" in the interval $[0, 0.3]$. For the frequent pattern extraction, we consider only the pipeline instances falling in these intervals. As to the support, 50% ensures that the pattern recurs on many of the explored instances and empirically showed to be a good threshold to have good efficiency in the extraction of frequent patterns.

Example 23 (Rules Recommendation)**.** *With reference to the Problem Graph in Example 22, the AutoML results are filtered according to the chosen metric, the algorithm [Raschka, 2018] is applied, and let us assume that the rule* `c3` *is recommended:*

[2] Maximal itemsets are patterns that are not contained in any other. For instance, given two frequent patterns, $\{a, b, c\}$ and $\{a, b\}$, the former is maximal while the latter is not.

```
c3 : ⇒ mandatory(⟨𝒟⟩, 𝒟t).
```

The constraints specifies "mandatory $\mathcal{D}$ in pipelines with $\mathcal{D}t$". As a matter of fact, it is well known that Discretization improves the performance of tree-based algorithms giving to them the ability to apply multiple split in the decision nodes. Figure 7.3c shows the effect of the applied constraint: a new portion of the search space is excluded from the extension ($\{p1\}$).

7.3 Experimental Evaluation

The performance of HAMLET depends on (i) the rules encoded in the LogicalKB and (ii) the rules recommended after each run. To test both the effectiveness and efficiency of our approach, we define three experimental settings.

- PKB (Preliminary Knowledge Base), HAMLET starts with a preliminary LogicalKB constraining the search space from the first iteration, and no rule mining is applied. The preliminary LogicalKB consists of the rules discovered in [Giovanelli et al., 2021a] and some well-known from the literature (e.g., suggested by scikit-learn[3]). The complete knowledge base can be found in the Github repository.

- IKA (Iterative Knowledge Augmentation), HAMLET starts with an empty LogicalKB, and all the rules recommended after each run are applied to extend the LogicalKB.

- PKB+IKA, HAMLET starts with a preliminary LogicalKB, and the rules recommended after each run are applied to extend the LogicalKB.

HAMLET run 4 times in every setting – intuitively, four runs of knowledge augmentation – the budget assigned to each run is 125 pipeline instances in 900 seconds (15 minutes). We also test against a baseline: we let AutoML explore 500 pipeline instances ($= 125 \cdot 4$) in a single run with a time budget of 3600 seconds ($= 900 \cdot 4$; 1 hour).

[3] https://scikit-learn.org/stable/auto_examples/preprocessing/plot_discretization.html

Table 7.2: Algorithms and number of hyperparameters for each of the steps in HAMLET. Algorithm names and hyperparameters are imported from the scikit-learn Python library.

Step	**Algorithm**	**#Hyperparameters**
Imputation	SimpleImputer	1
	IterativeImputer	2
Normalization	StandardScaler	2
	MinMaxScaler	0
	RobustScaler	2
	PowerTransformer	0
Discretization	Binarizer	1
	KBinsDiscretizer	3
Feature Eng.	SelectKBest	1
	PCA	1
Rebalancing	NearMiss	1
	SMOTE	1
Classification	DecisionTreeClassifier	7
	KNeighborsClassifier	3
	RandomForestClassifier	7
	AdaBoostClassifier	2
	MLPClassifier	6

For such an evaluation, we derive a search space out of 6 steps, 5 Data Preprocessing steps (Imputation, Normalization, Discretization, Feature Engineering, and Rebalancing) followed by the final Classification task. Since the tests are run on datasets from OpenML [Vanschoren et al., 2013] – a well-known repository for data acquisition and benchmarking – and it provides already-encoded datasets, we do not consider the encoding step. Except for that, we included all the Data Preprocessing steps and algorithms available in the scikit-learn [Pedregosa et al., 2011] Python library (plus imbalance-learn [Lemaître et al., 2017] for Rebalancing transformations). The leveraged steps, algorithms per step, and hyperparameters per algorithm are reported in Table 7.2.

The OpenML-CC18 suite is a well-known collection of 72 datasets for benchmarking. Given the time-consuming computation of each dataset (8 hours per dataset = 2 hours for the baseline + 6 hours for HAMLET in the three settings) – in this preliminary evaluation – we select a representative subset of datasets ac-

Table 7.3: Dataset descriptions.

OpenMLID [a]	Dataset	Instances		Features		Classes	
40983	wilt	4839	L	6	L	2	L
40499	texture	5500	L	41	L	11	H
1485	madelon	2600	L	501	H	2	L
1478	har	10229	L	562	H	6	H
1590	adult	48842	H	9	L	2	L
–	–	–	H	–	L	–	H
–	–	–	H	–	H	–	L
554	mnist_784	70000	H	785	H	10	H

[–] Not Applicable

[H] The value v is high for the meta-feature F if $v \geq \frac{1}{|F|} \sum_{f \in F} f$

[L] The value v is low for the meta-feature F if $v < \frac{1}{|F|} \sum_{f \in F} f$

[a] Datasets are available at `https://www.openml.org/d/<OpenMLID>`

cording to three meta-features provided by OpenML: number of instances, number of features, and number of classes. For each of the considered meta-features, we search for datasets with either high or low values, and we select the representatives that maximize the overall dataset diversification. Table 7.3 illustrates the 6 datasets that have been identified; note that some combinations of meta-features have no representative dataset in the suite. Among these, we do not report the results for the dataset mnist_784 since the number of explored pipeline instances is insufficient to validate the result (i.e., due to the time necessary to run a single pipeline instance, only 50 instances were explored out of 1000).

7.3.1 Effectiveness

We employ balanced accuracy as the quality metric. For instance, in case of (two) binary classes, such a score is

$$\texttt{Balanced accuracy} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right)$$

where TP and TN stand respectively for True Positive and True Negative (i.e., number of instances that have been correctly assigned to the positive and negative

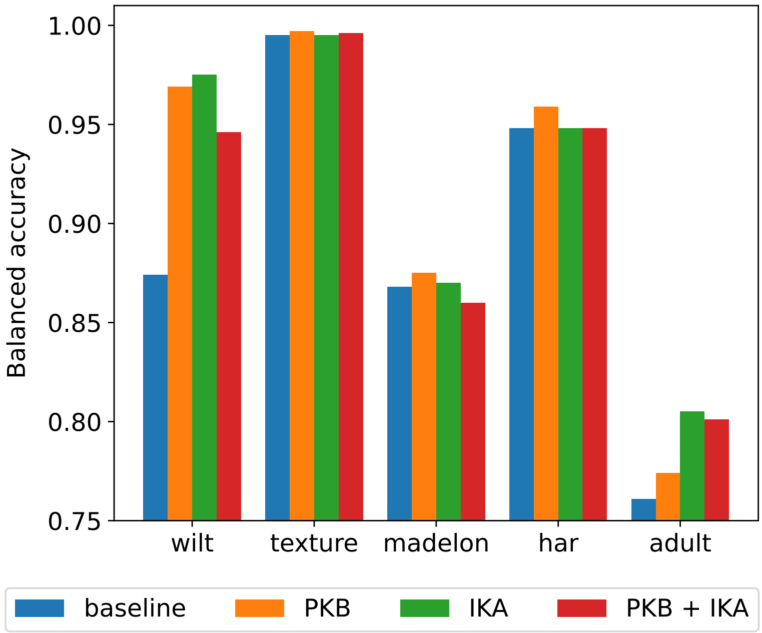


Figure 7.5: Results assessing the effectiveness of HAMLET w.r.t. the baseline.

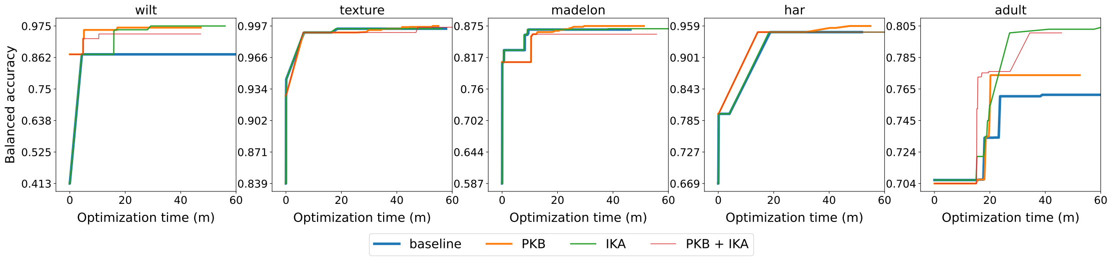


Figure 7.6: Results assessing the performance of HAMLET through the optimization time.

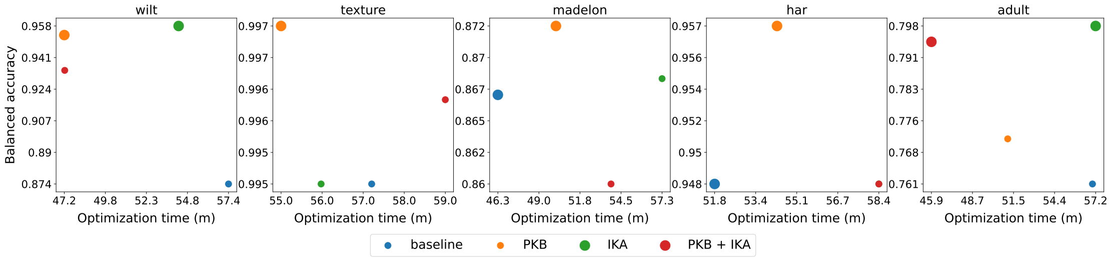


Figure 7.7: Comparison of the best pipeline instances characterized by optimization time and (balanced) accuracy, bigger circles represent settings that dominate the others.

classes), and FP and FN stand respectively for False Positive and False Negative (i.e., number of instances that have been mistakenly assigned to the positive and negative classes). The formulation generalized to more than 2 classes can be found at [Brodersen et al., 2010]. The score avoids inflated performance estimations on imbalanced datasets. For balanced datasets, the score is equal to the conventional accuracy (i.e., the number of correct predictions divided by the total number of predictions), otherwise it drops to $\frac{1}{\#classes}$.

Figure 7.5 illustrates the performance achieved by the baseline and the three settings of HAMLET. HAMLET is clearly beneficial since in all datasets the framework overcomes the baseline. The preliminary results highlight that both the LogicalKB and rule recommendation play important roles:

- When we warm-start the exploration with a non-empty LogicalKB (PKB), in all datasets HAMLET overcomes the baseline.
- When we only leverage rule recommendation (IKA), we achieve results that are better than or equivalent to PKB, indeed we are injecting in the LogicalKB new rules that are tailored to the dataset.
- The synergy of PKB+IKA performs better than PKB in adult, worse in wilt, and the two are comparable in the other datasets. On the one hand, the PKB act as a warm start mechanism that speeds up the optimization; on the other hand, if not aligned with the recommended rules, it can mitigate the benefits of IKA. This proves to be a promising direction that further requires investigation since merging the words will require further studies. Indeed, it is worth noting that the recommended rules can be overlapping with the ones in the LogicalKB, highlighting the need to improve the recommendation process by also considering the rules that are already present in the LogicalKB.

In PKB+IKA, IKA can introduce rules that contradict the ones in the LogicalKB of PKB; for instance when the PKB contains rules that are not "representative" of the dataset/algorithms in use. We believe that this is an added value of HAMLET since "incomplete" (or even wrong) LogicalKBs can be corrected/refined by a data-driven approach. Finally, PKB+IKA and IKA are likely to

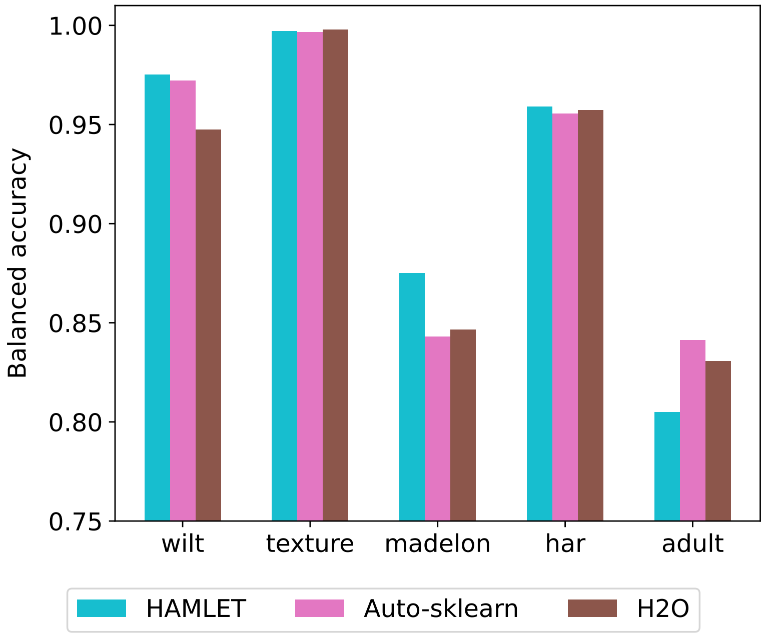


Figure 7.8: Results assessing the performance of HAMLET w.r.t. Auto-sklearn [Feurer et al., 2019] and H2O [LeDell and Poirier, 2020].

produce different rules, since in PKB+IKA the LogicalKB biases the exploration of the search space from the beginning (acting as a warm start mechanism).

Figure 7.8 compares HAMLET against two well-known AutoML frameworks: Auto-sklearn [Feurer et al., 2019] and H2O [LeDell and Poirier, 2020]. In four datasets out of five, HAMLET outperforms or is comparable to the two frameworks. Additionally, the added value of HAMLET is *explainability*. Hamlet is a human-in-the-loop AutoML framework tailored to the needs of DS that (i) enables the injection of their experience into the exploration process as well as (ii) the spreading and sharing of knowledge bases that encode what DSs have understood by the optimization of their pipelines.

7.3.2 Efficiency

Figure 7.6 shows how settings converge to the optimal pipeline instance. Noticeably, PKB and PKB+IKA start with higher accuracy than IKA and the baseline in four datasets out of five, proving how the preliminary LogicalKB warm starts

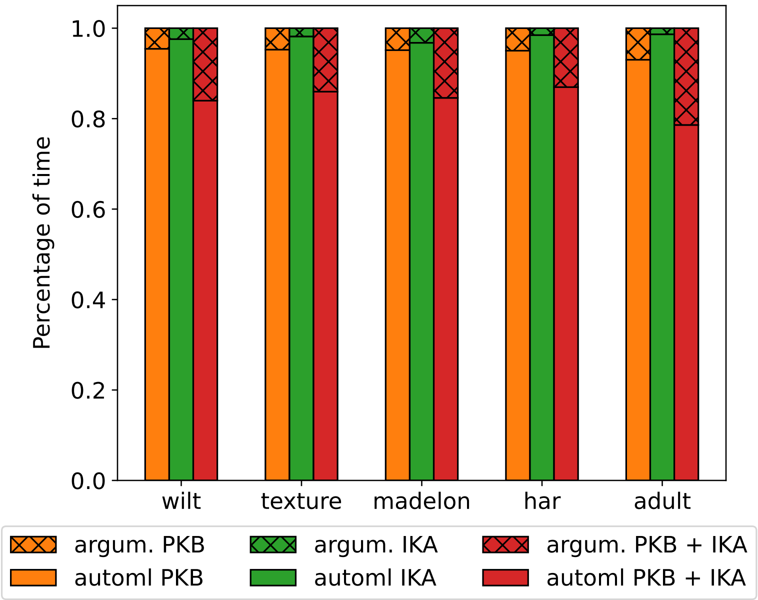


Figure 7.9: Computational time of the argumentation and AutoML processes.

the exploration. However, time and #iterations alone are not fair metrics for comparison; for instance, an optimization strategy could privilege simple algorithms taking small amounts of computational time but producing worse results than "more complex" algorithms. In the direction of multi-objective optimization (exploration time should be minimized while accuracy should be maximized), Figure 7.7 depicts which settings dominate the others using the Skyline operator [Borzsony et al., 2001]. A setting dominates another one if it is as good or better in all dimensions (time and accuracy) and better in at least one dimension (time or accuracy). PKB dominates in 80% of the datasets, IKA in 40%, PKB+IKA in 20%, and the baseline in 40%. Noticeably, the baseline is selected as dominating only in madelon and har datasets due to the fact that converges faster than HAMLET (although it converges to a pipeline instance with lower accuracy).

Finally, Figure 7.9 depicts the overhead introduced by the argumentation framework in HAMLET that, at maximum, is 20% of the computational time in the adult dataset. This proves that the argumentation time is marginal with respect to the duration of the optimization process. As expected, PKB+IKA shows the highest overhead since the number of rules to manage is the highest.

Chapter 8

Conclusions

As we approach the conclusion of this thesis, it is essential to revisit the goals that have guided this research and to reflect on the contributions made. At its inception, this thesis set out with the objective of enhancing the field of argumentation theory, particularly focusing on its application within the area of computational legal reasoning. One of the primary goals was also to bridge the gap between theoretical models of argumentation and their practical applications, ensuring that these models are not only academically robust but also practically applicable in real-world scenarios.

To achieve this, the thesis proposes several innovations. Firstly, it introduces significant advancements in the ASPIC$^+$ framework for meta-argumentation. This enhancement is crucial for enabling a more dynamic and nuanced representation of arguments, especially relevant in complex domains where arguments are often interwoven with intricate conflicts and interpretations. Also, the integration of the burden-of-persuasion concept into the structured meta-argumentation frameworks marks a significant stride in aligning computational models with the intricacies of legal processes.

Another cornerstone of this research is the development of the Arg2P framework. This technological innovation is designed to handle the nuances of argumentation in legal and AI contexts, managing inconsistent information and adapting to various argumentation needs. Moreover, its modularity and integration with logical programming make it a valuable tool in the realm of intelligent systems.

The final goal of this thesis, bridging argumentation and sub-symbolic AI, is epitomised in the selection of AutoML as the intersection area and the introduction of the HAMLET framework. This choice underscores the growing importance of making automated machine learning processes more interpretable and aligned with human reasoning, particularly in terms of ethical and legal considerations. This integration opens new avenues for the development of AI methodologies with a keen sense of responsibility, ensuring their application aligns with established standards and principles.

The contributions of this thesis hold particular significance in the context of legal reasoning, a domain where the precision and nuance of argumentation are paramount. The advancements in the ASPIC framework, with its enhanced capacity for meta-argumentation, bring a renewed level of sophistication to computational models of legal argumentation. This enhancement allows for a more intricate representation and analysis of legal arguments, closely mirroring the complexity and depth of human legal reasoning. The ability to encode conflicts, rules and preferences and develop meta-arguments within this framework aligns computational argumentation more closely with the multifaceted nature of legal discourse, where arguments are often layered and interdependent. Furthermore, the integration of the burden of persuasion concept, provides a more accurate and nuanced approach to modelling legal arguments, enabling AI systems to better replicate the decision-making processes found in human legal systems.

The Arg2P framework, as a practical embodiment of these theoretical advancements, demonstrates the feasibility of applying enhanced argumentation models in real-world legal scenarios. Its adaptability and capability to handle complex, inconsistent information make it a valuable tool for legal professionals and AI systems dealing with legal data. This framework can significantly aid in the processing, analysis, and simulation of legal arguments, paving the way for more sophisticated, AI-driven legal reasoning tools.

In the larger context of legal reasoning, all these contributions represent a step towards the development of AI systems that can effectively and ethically assist in legal decision-making processes. The enhanced argumentation models and tools introduced in this thesis provide a foundation for AI applications in law that are more transparent, reliable, and aligned with human legal reasoning. This alignment is crucial in an era where AI's role in legal processes is rapidly expanding, necessitating solutions that are both technologically advanced and deeply attuned to the nuances of legal practice.

While we have summarised the main contributions of this thesis, it is equally important to acknowledge the areas that remain open and present opportunities for future development.

The exploration of meta-argumentation within the ASPIC$^+$ framework sets a foundation for future research, aimed at expanding beyond the grounded semantics which is the primary focus of this investigation. Formal proofs provided herein are confined to grounded semantics, as delineated in the introduction, but the potential for expansion into other Dung's semantics is both vast and compelling. The enhancements introduced to the ASPIC framework have already imbued it with newfound flexibility, yet the possibilities for further expansion are extensive. The domain of discourse in argumentation is rich and varied, and this foray into the meta aspect is likely to unravel new questions, fostering deeper insights and solutions. It is an intriguing prospect to examine the applicability and suitability of these introduced mechanisms to broader and more diverse argumentative contexts.

On the technological front, the thesis delves into both algorithmic and technological aspects. In terms of algorithms, the focus is on enhancing the efficiency of the structured resolution process. Again, the primary focus of the work is centred on grounded semantics. While this approach aligns with the thesis's focus and proves sufficient within its scope, the potential for expansion to other classical semantics is evident and worthwhile. Moreover, the dynamic nature of the computer science landscape presents numerous opportunities for innovative development, particularly in integrating these tools more effectively in the IT world. This approach is not just about enhancing argumentation tools for academic exploration, but about making them practically viable and valuable in real-world IT scenarios.

On the same line, the Arg2P technology, as introduced in this thesis, represents only the foundational steps in what promises to be a significant journey in the field of argumentation in AI. While fundamental elements and some additional features have been implemented, this is merely the beginning of its developmental trajectory. The potential for further enhancement and expansion is vast, with the integration of both existing and new elements from the argumentation theoretical background being key to its evolution.

The theoretical landscape of argumentation is both rich and multifaceted, often finding its existence primarily in academic discourse. By bringing these theoretical concepts to life, even if it means compromising certain features or properties for efficiency, Arg2P can serve as a critical step in integrating argumentation more broadly within the AI panorama. This endeavour is not just about improving specific features of the framework; indeed, every aspect of Arg2P holds potential for enhancement. For each feature that has been implemented, there are likely numerous ways it could be refined or re-imagined.

However, the focus should not solely be on technological advancement but also on fostering a community around Arg2P. This community-building is crucial to ensure that Arg2P evolves beyond a prototype and becomes a widely used and continuously developed tool in the field of argumentation technologies. The aim is to avoid the fate of becoming another forgotten project, but rather to establish Arg2P as a cornerstone in the ongoing discourse and development of argumentation technologies. Building such a community would encourage collaborative improvement, diverse application, and sustained innovation, ensuring that the framework remains relevant and valuable in advancing the integration of argumentation in AI.

Moving to the last part of this thesis, AutoML is chosen as the focal point for this intersection due to its pivotal role in simplifying and automating the complex process of model selection, configuration, and optimisation in Machine Learning. However, the current iteration of HAMLET, while advanced in imposing technical constraints on the ML pipeline, does not yet fully address aspects such as hyperparameter optimisation. Expanding HAMLET to incorporate such features would not only enhance its power but also increase its practical applicability in a broader range of ML scenarios.

More intriguing, however, is the potential of HAMLET to encode more abstract constraints, particularly those related to ethical and regulatory preferences. The recent surge in research aimed at quantifying and evaluating the ethics of AI models offers a rich tapestry of criteria that can be integrated into AutoML frameworks. Embedding these ethical considerations into HAMLET could change the way Data

Scientists control the optimisation and development of ML models, ensuring that these models adhere not only to technical specifications but also to ethical and regulatory standards.

The legal implications of such an integration are profound. By incorporating ethical and regulatory constraints into the ML development process, HAMLET can ensure that the resulting models are not only efficient and accurate but also compliant with legal and ethical guidelines. This is particularly crucial in sectors where AI decisions have significant societal impacts, such as healthcare, finance, and public policy. The ability to encode and enforce these constraints within the AutoML process can lead to more responsible and trustworthy AI systems, aligning technological advancements with societal values and legal requirements.

In conclusion, this thesis embodies a significant effort to advance argumentation theory, especially within the realms of computational legal reasoning and the field of sub-symbolic AI. By enhancing the ASPIC$^+$ framework and pioneering the development of the Arg2P and HAMLET frameworks, this research has effectively bridged the gap between theoretical models and practical applications. More importantly, it has underscored the fundamental role of argumentation theory in the AI landscape, drawing a crucial link with the evolving technological panorama. Setting aside all the technical advancements and theoretical explorations, the foremost aspiration of this thesis is singular and clear: to contribute to the positioning of argumentation theory as an indispensable component in the responsible development and application of future AI systems. It is with this intention that the work presented here was undertaken, and it is with this hope that it is concluded. May this contribution serve as a meaningful step in the direction of realising a future where AI systems are not only advanced in their capabilities but also guided by the principles of sound reasoning and argumentation.

Bibliography

[Alsinet et al., 2010] Alsinet, T., Béjar, R., and Godo, L. (2010). A characterization of collective conflict for defeasible argumentation. In *Computational Models of Argument*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 27–38. IOS Press.

[Alsinet et al., 2012] Alsinet, T., Béjar, R., Godo, L., and Guitart, F. (2012). Using answer set programming for an scalable implementation of defeasible argumentation. In *IEEE 24th International Conference on Tools with Artificial Intelligence*, pages 1016–1021. IEEE Computer Society.

[Amgoud et al., 2004] Amgoud, L., Cayrol, C., and Lagasquie-Schiex, M. (2004). On the bipolarity in argumentation frameworks. In *10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 1–9.

[Andrighetto et al., 2013] Andrighetto, G., Governatori, G., Noriega, P., and van der Torre, L. W. (2013). *Normative multi-agent systems*, volume 4 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Arsénio et al., 2014] Arsénio, A., Serra, H., Francisco, R., Nabais, F., Andrade, J., and Serrano, E. (2014). Internet of Intelligent Things: Bringing artificial intelligence into things and communication networks. In *Inter-cooperative Collective Intelligence: Techniques and Applications*, volume 495 of *Studies in Computational Intelligence*, pages 1–37. Springer.

[Bao et al., 2017] Bao, Z., Cyras, K., and Toni, F. (2017). ABAplus: Attack reversal in abstract and structured argumentation with preferences. In *PRIMA 2017: Principles and Practice of Multi-Agent Systems*, volume 10621 of *Lecture Notes in Computer Science*, pages 420–437. Springer.

[Baroni et al., 2011a] Baroni, P., Caminada, M., and Giacomin, M. (2011a). An introduction to argumentation semantics. *Knowledge Engineering Review*, 26(4):365–410.

[Baroni et al., 2011b] Baroni, P., Caminada, M., and Giacomin, M. (2011b). An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410.

[Baroni et al., 2011c] Baroni, P., Caminada, M., and Giacomin, M. (2011c). An introduction to argumentation semantics. *The knowledge engineering review*, 26(4):365–410.

[Baroni et al., 2018] Baroni, P., Gabbay, D., Giacomin, M., and van der Torre, L. (2018). *Handbook of Formal Argumentation.* London, England: College Publications.

[Barringer et al., 2012] Barringer, H., Gabbay, D. M., and Woods, J. (2012). Modal and temporal argumentation networks. *Argument & Computation*, 3(2-3):203–227.

[Bench-Capon et al., 1993] Bench-Capon, T., Coenen, F., and Orton, P. (1993). Argument-based explanation of the British nationality act as a logic program. *Information and Communications Technology Law*, 2(1):53–66.

[Bench-Capon, 2002] Bench-Capon, T. J. M. (2002). Value-based argumentation frameworks. In *9th International Workshop on Non-Monotonic Reasoning (NMR 2002)*, pages 443–454.

[Besnard et al., 2014a] Besnard, P., Garcia, A., Hunter, A., Modgil, S., Prakken, H., Simari, G., and Toni, F. (2014a). Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4.

[Besnard et al., 2014b] Besnard, P., García, A. J., Hunter, A., Modgil, S., Prakken, H., Simari, G. R., and Toni, F. (2014b). Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4.

[Besnard et al., 2014c] Besnard, P., García, A. J., Hunter, A., Modgil, S., Prakken, H., Simari, G. R., and Toni, F. (2014c). Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4.

[Bistarelli and Santini, 2011] Bistarelli, S. and Santini, F. (2011). Conarg: A constraint-based computational framework for argumentation systems. In *IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 605–612. Institute of Electrical and Electronics Engineers.

[Boella et al., 2009a] Boella, G., Gabbay, D. M., van der Torre, L., and Villata, S. (2009a). Meta-argumentation modelling I: Methodology and techniques. *Studia Logica*, 93(2–3):297.

[Boella et al., 2009b] Boella, G., Gabbay, D. M., van der Torre, L. W. N., and Villata, S. (2009b). Meta-argumentation modelling I: methodology and techniques. *Studia Logica*, 93(2-3):297–355.

[Borzsony et al., 2001] Borzsony, S., Kossmann, D., and Stocker, K. (2001). The skyline operator. In *Proceedings 17th international conference on data engineering*, pages 421–430. IEEE.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

[Brodersen et al., 2010] Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *20th International Conference on Pattern Recognition, ICPR 2010, Istanbul, Turkey, 23-26 August 2010*, pages 3121–3124. IEEE Computer Society.

[Bryant et al., 2006] Bryant, D., Krause, P. J., and Vreeswijk, G. (2006). Argue tuProlog: A lightweight argumentation engine for agent applications. In *Computational Models of Argument*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 27–32. IOS Press.

[Calegari, 2018] Calegari, R. (2018). *Micro-Intelligence for the IoT: Logic-based Models and Technologies*. PhD thesis, Alma Mater Studiorum—Università di Bologna, Bologna, Italy.

[Calegari et al., 2021a] Calegari, R., Ciatto, G., Mascardi, V., and Omicini, A. (2021a). Logic-based technologies for multi-agent systems: A systematic literature review. *Autonomous Agents and Multi-Agent Systems*, 35(1):1:1–1:67.

[Calegari et al., 2019] Calegari, R., Contissa, G., Lagioia, F., Omicini, A., and Sartor, G. (2019). Defeasible systems in legal reasoning: A comparative assessment. In Araszkiewicz, M. and Rodríguez-Doncel, V., editors, *Legal Knowledge and Information Systems. JURIX 2019: The Thirty-second Annual Conference*, volume 322 of *Frontiers in Artificial Intelligence and Applications*, pages 169–174. IOS Press.

[Calegari et al., 2021b] Calegari, R., Omicini, A., and Sartor, G. (2021b). Explainable and ethical AI: A perspective on argumentation and logic programming. In Baldoni, M. and Bandini, S., editors, *AIxIA 2020 – Advances in Artificial Intelligence*, volume 12414 of *Lecture Notes in Computer Science*, pages 19–36. Springer Nature.

[Calegari et al., 2021c] Calegari, R., Pisano, G., Omicini, A., and Sartor, G. (2021c). Arg2P: An argumentation framework for explainable intelligent systems. *Journal of Logic and Computation.*

[Calegari et al., 2021d] Calegari, R., Riveret, R., and Sartor, G. (2021d). The burden of persuasion in structured argumentation. In Maranhão, J. and Wyner, A. Z., editors, *ICAIL'21: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*, ICAIL'21, pages 180–184. ACM.

[Calegari and Sartor, 2020a] Calegari, R. and Sartor, G. (2020a). Burden of persuasion in argumentation. In Ricca, F., Russo, A., Greco, S., Leone, N., Artikis, A., Friedrich, G., Fodor, P., Kimmig, A., Lisi, F., Maratea, M., Mileo, A., and Riguzzi, F., editors, *36th International Conference on Logic Programming (ICLP 2020)*, volume 325 of *Electronic Proceedings in Theoretical Computer Science*, pages 151–163. Open Publishing Association.

[Calegari and Sartor, 2020b] Calegari, R. and Sartor, G. (2020b). A model for the burden of persuasion in argumentation. In Villata, S., Harašta, J., and Křemen, P., editors, *Legal Knowledge and Information Systems. JURIX 2020: The Thirty-third Annual Conference*, volume 334 of *Frontiers in Artificial Intelligence and Applications*, pages 13–22, Brno, Czech Republic. IOS Press.

[Caminada, 2015] Caminada, M. (2015). A discussion game for grounded semantics. In *Theory and Applications of Formal Argumentation*, volume 9524 of *Lecture Notes in Computer Science*, pages 59–73. Springer.

[Caminada and Amgoud, 2007a] Caminada, M. and Amgoud, L. (2007a). On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5—6):286–310.

[Caminada and Amgoud, 2007b] Caminada, M. and Amgoud, L. (2007b). On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5—6):286–310.

[Caminada and Amgoud, 2007c] Caminada, M. and Amgoud, L. (2007c). On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5–6):286–310.

[Caminada and Uebis, 2020] Caminada, M. and Uebis, S. (2020). An implementation of argument-based discussion using ASPIC-. In *Computational Models of Argument*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, pages 455–456. IOS Press.

[Carrera and Iglesias, 2015] Carrera, Á. and Iglesias, C. A. (2015). A systematic review of argumentation techniques for multi-agent systems research. *Artificial Intelligence Review*, 44(4):509–535.

[Ciatto et al., 2021] Ciatto, G., Calegari, R., and Omicini, A. (2021). Lazy stream manipulation in Prolog via backtracking: The case of 2p-kt:. In Faber, W., Friedrich, G., Gebser, M., and Morak, M., editors, *Logics in Artificial Intelligence*, volume 12678 of *Lecture Notes in Computer Science*, pages 407–420. Springer. 17th European Conference, JELIA 2021, Virtual Event, May 17–20, 2021, Proceedings.

[Ciatto et al., 2020] Ciatto, G., Calegari, R., Siboni, E., Denti, E., and Omicini, A. (2020). 2P-Kt: logic programming with objects & functions in Kotlin. In Calegari, R., Ciatto, G., Denti, E., Omicini, A., and Sartor, G., editors, *WOA 2020 – 21th Workshop "From Objects to Agents"*, volume 2706 of *CEUR Workshop Proceedings*, pages 219–236, Aachen, Germany. Sun SITE Central Europe, RWTH Aachen University.

[Cossentino et al., 2018] Cossentino, M., Lopes, S., Nuzzo, A., Renda, G., and Sabatucci, L. (2018). A comparison of the basic principles and behavioural aspects of Akka, JaCaMo and Jade development frameworks. In *Proceedings of the 19th Workshop "From Objects to Agents"*, volume 2215 of *CEUR Workshop Proceedings*, pages 133–141. CEUR-WS.org.

[Craven and Toni, 2016] Craven, R. and Toni, F. (2016). Argument graphs and assumption-based argumentation. *Artificial Intelligence*, 233:1–59.

[Craven et al., 2013] Craven, R., Toni, F., and Williams, M. (2013). Graph-based dispute derivations in assumption-based argumentation. In *Theory and Applications of Formal Argumentation*, volume 8306 of *Lecture Notes in Computer Science*, pages 46–62. Springer.

[Crisan and Fiore-Gartland, 2021] Crisan, A. and Fiore-Gartland, B. (2021). Fits and starts: Enterprise use of automl and the role of humans in the loop. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

[da Costa Pereira et al., 2017] da Costa Pereira, C., Tettamanzi, A. G. B., Liao, B., Malerba, A., Rotolo, A., and van der Torre, L. W. N. (2017). Combining fuzzy logic and formal argumentation for legal interpretation. In *Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law, ICAIL 2017, London, United Kingdom, June 12-16, 2017*, pages 49–58.

[Denti et al., 2005] Denti, E., Omicini, A., and Ricci, A. (2005). Multi-paradigm Java-Prolog integration in tuProlog. *Science of Computer Programming*, 57(2):217–250.

[Drozdal et al., 2020] Drozdal, J., Weisz, J., Wang, D., Dass, G., Yao, B., Zhao, C., Muller, M., Ju, L., and Su, H. (2020). Trust in automl: exploring information needs for establishing trust in automated machine learning systems. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 297–307.

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

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

[Dung and Thang, 2018] Dung, P. M. and Thang, P. M. (2018). Fundamental properties of attack relations in structured argumentation with priorities. *Artificial Intelligence*, 255:1–42.

[Dung et al., 2019] Dung, P. M., Thang, P. M., and Son, T. C. (2019). On structured argumentation with conditional preferences. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 2792–2800, Honolulu, Hawaii, USA. AAAI Press.

[Dvorák et al., 2020] Dvorák, W., Rapberger, A., Wallner, J. P., and Woltran, S. (2020). ASPARTIX-V19 - an answer-set programming based system for abstract argumentation. In *Foundations of Information and Knowledge Systems—11th International Symposium*, volume 12012 of *Lecture Notes in Computer Science*, pages 79–89. Springer.

[Falkner et al., 2018] Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR.

[Feurer et al., 2019] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., and Hutter, F. (2019). Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*, pages 113–134. Springer, Cham.

[Francia et al., 2022] Francia, M., Gallinucci, E., and Golfarelli, M. (2022). COOL: A framework for conversational OLAP. *Inf. Syst.*, 104:101752.

[Francia et al., 2019] Francia, M., Golfarelli, M., and Rizzi, S. (2019). Augmented business intelligence. In Song, I., Romero, O., and Wrembel, R., editors, *Proceedings of the 21st International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT Joint Conference, DOLAP@EDBT/ICDT 2019, Lisbon, Portugal, March 26, 2019*, volume 2324 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Frazier, 2018] Frazier, P. I. (2018). A tutorial on bayesian optimization. *CoRR*, abs/1807.02811.

[Gabbay, 2009] Gabbay, D. M. (2009). Semantics for higher level attacks in extended argumentation frames part 1: Overview. *Studia Logica*, 93(2-3):357–381.

[García et al., 2020] García, A. J., Prakken, H., and Simari, G. R. (2020). A comparative study of some central notions of ASPIC+ and DeLP. *Theory and Practice of Logic Programming*, 20(3):358–390.

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

[Gil et al., 2019] Gil, Y., Honaker, J., Gupta, S., Ma, Y., D'Orazio, V., Garijo, D., Gadewar, S., Yang, Q., and Jahanshad, N. (2019). Towards human-guided machine learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 614–624.

[Giovanelli et al., 2021a] Giovanelli, J., Bilalli, B., and Abelló, A. (2021a). Data pre-processing pipeline generation for autoetl. *Information Systems*, page 101957.

[Giovanelli et al., 2021b] Giovanelli, J., Bilalli, B., and Abelló, A. (2021b). Effective data pre-processing for automl. In *Proceedings of the 23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP)*, volume 2840 of *CEUR Workshop Proceedings*, pages 1–10. CEUR-WS.org.

[Golovin et al., 2017] Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. (2017). Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495.

[Gordon et al., 2009] Gordon, T. F., Governatori, G., and Rotolo, A. (2009). Rules and norms: Requirements for rule interchange languages in the legal domain. In Governatori, G., Hall, J., and Paschke, A., editors, *Rule Interchange and Applications, International Symposium, RuleML 2009*, volume 5858 of *Lecture Notes in Computer Science*, pages 282–296. Springer.

[Gordon et al., 2007] Gordon, T. F., Prakken, H., and Walton, D. (2007). The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10-15):875–896.

[Gordon and Walton, 2016] Gordon, T. F. and Walton, D. (2016). Formalizing balancing arguments. In *Computational Models of Argument*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 327–338. IOS Press.

[Harrison and Rubinfeld, 1978] Harrison, D. and Rubinfeld, D. (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5:81–102.

[Hewitt et al., 1973] Hewitt, C., Bishop, P. B., and Steiger, R. (1973). A universal modular ACTOR formalism for artificial intelligence. In *3rd International Joint Conference on Artificial Intelligence*, pages 235–245. William Kaufmann.

[Hulstijn and van der Torre, 2004] Hulstijn, J. and van der Torre, L. W. (2004). Combining goal generation and planning in an argumentation framework. In *International Workshop on Non-monotonic Reasoning (NMR'04)*, pages 212–218.

[Jung et al., 2001] Jung, H., Tambe, M., and Kulkarni, S. (2001). Argumentation as distributed constraint satisfaction: Applications and results. In *5th International Conference on Autonomous Agents (Agents '01)*, pages 324–331.

[Khuat et al., 2022] Khuat, T. T., Kedziora, D. J., and Gabrys, B. (2022). The roles and modes of human interactions with automated machine learning systems. *arXiv preprint arXiv:2205.04139*.

[Kok et al., 2012] Kok, E. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. (2012). Testing the benefits of structured argumentation in multi-agent deliberation dialogues. In *11th International Conference on Autonomous Agents and Multiagent Systems*, volume 3, pages 1411–1412, Valencia, Spain. International Foundation for Autonomous Agents and Multiagent Systems.

[Kotthoff et al., 2019] Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2019). Auto-weka: Automatic model selection and hyperparameter optimization in weka. In *Automated Machine Learning*, pages 81–95. Springer, Cham.

[Kraska et al., 2013] Kraska, T., Talwalkar, A., Duchi, J. C., Griffith, R., Franklin, M. J., and Jordan, M. I. (2013). Mlbase: A distributed machine-learning system. In *Cidr*, volume 1, pages 2–1.

[Krippendorff, 2004] Krippendorff, K. (2004). Intrinsic motivation and human-centred design. *Theoretical Issues in Ergonomics Science*, 5(1):43–72.

[Kröll et al., 2017a] Kröll, M., Pichler, R., and Woltran, S. (2017a). On the complexity of enumerating the extensions of abstract argumentation frameworks. In Sierra, C., editor, *26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 1145–1152. IJCAI.org.

[Kröll et al., 2017b] Kröll, M., Pichler, R., and Woltran, S. (2017b). On the complexity of enumerating the extensions of abstract argumentation frameworks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 1145–1152, Melbourne, Australia. ijcai.org.

[LeDell and Poirier, 2020] LeDell, E. and Poirier, S. (2020). H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020.

[Lee and Macke, 2020] Lee, D. J.-L. and Macke, S. (2020). A human-in-the-loop perspective on automl: Milestones and the road ahead. *IEEE Data Engineering Bulletin*.

[Lehtonen et al., 2017] Lehtonen, T., Wallner, J. P., and Järvisalo, M. (2017). From structured to abstract argumentation: Assumption-based acceptance via AF reasoning. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 10369 of *Lecture Notes in Computer Science*, pages 57–68. Springer.

[Lehtonen et al., 2020] Lehtonen, T., Wallner, J. P., and Järvisalo, M. (2020). An answer set programming approach to argumentative reasoning in the ASPIC+ framework. In *17th International Conference on Principles of Knowledge Representation and Reasoning*, pages 636–646.

[Lemaître et al., 2017] Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

[Modgil and Bench-Capon, 2011] Modgil, S. and Bench-Capon, T. J. M. (2011). Metalevel argumentation. *Journal of Logic and Computation*, 21:959–1003.

[Modgil and Caminada, 2009] Modgil, S. and Caminada, M. (2009). Proof theories and algorithms for abstract argumentation frameworks. In *Argumentation in Artificial Intelligence*, pages 105–129. Springer.

[Modgil and Prakken, 2010] Modgil, S. and Prakken, H. (2010). Reasoning about preferences in structured extended argumentation frameworks. In Baroni, P., Cerutti, F., Giacomin, M., and Simari, G., editors, *Computational Models of Argument. Proceedings of COMMA 2010*, pages 347–58. IOS.

[Modgil and Prakken, 2013] Modgil, S. and Prakken, H. (2013). A general account of argumentation with preferences. *Artificial Intelligence*, 195:361–397.

[Modgil and Prakken, 2014a] Modgil, S. and Prakken, H. (2014a). The $ASPIC^+$ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62.

[Modgil and Prakken, 2014b] Modgil, S. and Prakken, H. (2014b). The $ASPIC^+$ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62.

[Müller et al., 2013] Müller, J., Hunter, A., and Taylor, P. (2013). Meta-level argumentation with argument schemes. In *International Conference on Scalable Uncertainty Management*, volume 8078 of *Lecture Notes in Computer Science*, pages 92–105, Washington, DC, USA. Springer, Springer.

[Muller et al., 2019] Muller, M., Lange, I., Wang, D., Piorkowski, D., Tsay, J., Liao, Q. V., Dugan, C., and Erickson, T. (2019). How data science workers work with data: Discovery, capture, curation, design, creation. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–15.

[Niskanen and Järvisalo, 2020] Niskanen, A. and Järvisalo, M. (2020). μ-toksia: An efficient abstract argumentation reasoner. In *17th International Conference on Principles of Knowledge Representation and Reasoning*, pages 800–804.

[Nute, 2001] Nute, D. (2001). Defeasible logic. In *Proceedings of the 14th International Conference on Applications of Prolog, INAP 2001*, pages 87–114. The Prolog Association of Japan.

[Ogunniye et al., 2018] Ogunniye, G., Toniolo, A., and Oren, N. (2018). Meta-argumentation frameworks for multi-party dialogues. In *International Conference on Principles and Practice of Multi-Agent Systems*, volume 11224 of *Lecture Notes in Computer Science*, pages 585–593, Tokyo, Japan. Springer, Springer.

[Oliva et al., 2008] Oliva, E., McBurney, P., and Omicini, A. (2008). Co-argumentation artifact for agent societies. In Parsons, S., Rahwan, I., and Reed, C., editors, *Argumentation in Multi-Agent Systems*, volume 4946 of *Lecture Notes in Computer Science*, chapter 3, pages 31–46. Springer. 4th International Workshop (ArgMAS 2007), Honolulu, HI, USA, 15 May 2007. Revised Selected and Invited Papers.

[Oliva et al., 2009] Oliva, E., Viroli, M., Omicini, A., and McBurney, P. (2009). Argumentation and artifact for dialog support. In Rahwan, I. and Moraitis, P., editors, *Argumentation in Multi-Agent Systems*, volume 5384 of *LNAI*, chapter 7, pages 107–121. Springer. 4th International Workshop (ArgMAS 2008), Estoril, Portugal, 12 May 2008. Revised Selected and Invited Papers.

[Olivieri et al., 2021] Olivieri, F., Governatori, G., Cristani, M., and Sattar, A. (2021). Computing defeasible meta-logic. In *Logics in Artificial Intelligence*, pages 69–84, Cham. Springer International Publishing.

[Omicini and Calegari, 2019] Omicini, A. and Calegari, R. (2019). Injecting (micro)intelligence in the IoT: Logic-based approaches for (M)MAS. In Lin, D., Ishida, T., Zambonelli, F., and Noda, I., editors, *Massively Multi-Agent Systems II*, volume 11422 of *Lecture Notes in Computer Science*, chapter 2, pages 21–35. Springer.

[Omicini and Mariani, 2013] Omicini, A. and Mariani, S. (2013). Agents & multiagent systems: En route towards complex intelligent systems. *Intelligenza Artificiale*, 7(2):153–164.

[Ono et al., 2021] Ono, J. P., Castelo, S., Lopez, R., Bertini, E., Freire, J., and Silva, C. T. (2021). Pipelineprofiler: A visual analytics tool for the exploration of automl pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):390–400.

[Ossowski, 2012] Ossowski, S., editor (2012). *Agreement Technologies*, volume 8 of *Law, Governance and Technology Series*. Springer Netherlands.

[Paulson, 2018] Paulson, L. C. (2018). Computational logic: its origins and applications. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2210).

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Peng et al., 2020] Peng, D., Dong, X., Real, E., Tan, M., Lu, Y., Bender, G., Liu, H., Kraft, A., Liang, C., and Le, Q. (2020). Pyglove: Symbolic programming for automated machine learning. *Advances in Neural Information Processing Systems*, 33:96–108.

[Perrone et al., 2021] Perrone, V., Donini, M., Zafar, M. B., Schmucker, R., Kenthapadi, K., and Archambeau, C. (2021). Fair bayesian optimization. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 854–863.

[Pfisterer et al., 2019] Pfisterer, F., Thomas, J., and Bischl, B. (2019). Towards human centered automl. *arXiv preprint arXiv:1911.02391*.

[Pisano et al., 2020] Pisano, G., Calegari, R., Omicini, A., and Sartor, G. (2020). Arg-tuProlog: A tuProlog-based argumentation framework. In Calimeri, F., Perri, S., and Zumpano, E., editors, *CILC 2020 – Italian Conference on Computational Logic. Proceedings of the 35th Italian Conference on Computational Logic*, volume 2710 of *CEUR Workshop Proceedings*, pages 51–66, Aachen, Germany. Sun SITE Central Europe, RWTH Aachen University, CEUR-WS.

[Pisano et al., 2021] Pisano, G., Calegari, R., Omicini, A., and Sartor, G. (2021). A mechanism for reasoning over defeasible preferences in Arg2P. In Monica, S. and Bergenti, F., editors, *CILC 2021 – Italian Conference on Computational Logic. Proceedings of the 36th Italian Conference on Computational Logic*, volume 3002 of *CEUR Workshop Proceedings*, pages 16–30, Parma, Italy. CEUR-WS.

[Podlaszewski et al., 2011] Podlaszewski, M., Caminada, M., and Pigozzi, G. (2011). An implementation of basic argumentation components. In *10th International Conference on Autonomous Agents and Multiagent Systems*, volume 3, pages 1307–1308. International Foundation for Autonomous Agents and Multiagent Systems.

[Prakken, 2005] Prakken, H. (2005). Ai & law, logic and argument schemes. *Argumentation*, 19(3):303–320.

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

[Prakken and Sartor, 1996] Prakken, H. and Sartor, G. (1996). A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law*, 4:331–68.

[Quemy, 2019] Quemy, A. (2019). Data pipeline selection and optimization. In *DOLAP*.

[Raschka, 2018] Raschka, S. (2018). Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24).

[Rienstra et al., 2011] Rienstra, T., Perotti, A., Villata, S., Gabbay, D. M., and van der Torre, L. W. N. (2011). Multi-sorted argumentation. In Modgil, S., Oren, N., and Toni, F., editors, *Theorie and Applications of Formal Argumentation – First International Workshop, TAFA 2011*, volume 7132 of *Lecture Notes in Computer Science*, pages 215–231. Springer.

[Riveret et al., 2019] Riveret, R., Rotolo, A., and Sartor, G. (2019). A deontic argumentation framework towards doctrine reification. *Journal of Applied Logics—IfCoLog Journal of Logics and their Applications*, 6(5):903–940. Special Issue: Reasoning for Legal AI.

[Rotolo et al., 2015] Rotolo, A., Governatori, G., and Sartor, G. (2015). Deontic defeasible reasoning in legal interpretation: Two options for modelling interpretive arguments. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law (ICAIL'05)*, pages 99–108. ACM.

[Snaith and Reed, 2012] Snaith, M. and Reed, C. (2012). TOAST: online ASPIC^{+} implementation. In *Computational Models of Argument*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 509–510. IOS Press.

[Srikant and Agrawal, 1995] Srikant, R. and Agrawal, R. (1995). Mining generalized association rules.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *International conference on extending database technology*, pages 1–17. Springer.

[Tan, 2017] Tan, H. (2017). A brief history and technical review of the expert system research. *IOP Conference Series: Materials Science and Engineering*, 242.

[Toni, 2013] Toni, F. (2013). A generalised framework for dispute derivations in assumption-based argumentation. *Artificial Intelligence*, 195:1–43.

[Toni, 2014] Toni, F. (2014). A tutorial on assumption-based argumentation. *Argument & Computation*, 5(1):89–117.

[van Gijzel and Prakken, 2012] van Gijzel, B. and Prakken, H. (2012). Relating Carneades with abstract argumentation via the ASPIC^{+} framework for structured argumentation. *Argument & Computation*, 3(1):21–47.

[Vanschoren et al., 2013] Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). Openml: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

[Vartak et al., 2015] Vartak, M., Ortiz, P., Siegel, K., Subramanyam, H., Madden, S., and Zaharia, M. (2015). Supporting fast iteration in model building. In *NIPS Workshop LearningSys*, pages 1–6.

[Vasconcelos et al., 2002] Vasconcelos, W. W., Sabater, J., Sierra, C., and Querol, J. (2002). Skeleton-based agent development for electronic institutions. In *1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2 (AAMAS '02)*, pages 696–703, New York, NY, USA. ACM.

[Vreeswijk, 1997a] Vreeswijk, G. (1997a). Abstract argumentation systems. *Artif. Intell.*, 90(1-2):225–279.

[Vreeswijk, 1997b] Vreeswijk, G. (1997b). Abstract argumentation systems. *Artificial Intelligence*, 90(1–2):225–279.

[Walton et al., 2021] Walton, D., Macagno, F., and Sartor, G. (2021). *Statutory Interpretation. Pragmatics and Argumentation.* Cambridge University Press.

[Walton et al., 2008] Walton, D., Reed, C., and Macagno, F. (2008). *Argumentation Schemes.* Cambridge University Press, United Kingdom.

[Wang et al., 2021a] Wang, C., Wu, Q., Weimer, M., and Zhu, E. (2021a). Flaml: A fast and lightweight automl library. *Proceedings of Machine Learning and Systems*, 3:434–447.

[Wang et al., 2021b] Wang, D., Andres, J., Weisz, J. D., Oduor, E., and Dugan, C. (2021b). Autods: Towards human-centered automation of data science. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

[Wang et al., 2020] Wang, D., Ram, P., Weidele, D. K. I., Liu, S., Muller, M., Weisz, J. D., Valente, A., Chaudhary, A., Torres, D., Samulowitz, H., et al. (2020). Autoai: Automating the end-to-end ai lifecycle with humans-in-the-loop. In *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*, pages 77–78.

[Wang et al., 2019a] Wang, D., Weisz, J. D., Muller, M., Ram, P., Geyer, W., Dugan, C., Tausczik, Y., Samulowitz, H., and Gray, A. (2019a). Human-ai collaboration in data science: Exploring data scientists' perceptions of automated ai. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–24.

[Wang et al., 2019b] Wang, Q., Ming, Y., Jin, Z., Shen, Q., Liu, D., Smith, M. J., Veeramachaneni, K., and Qu, H. (2019b). Atmseer: Increasing transparency and controllability in automated machine learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

[Wang et al., 2022] Wang, X., Hosseininasab, A., Colunga, P., Kadioglu, S., and van Hoeve, W.-J. (2022). Seq2pat: Sequence-to-pattern generation for constraint-based sequential pattern mining. *Proceedings of the AAAI Conference on Artificial Intelligence*, TBD(TBD):TBD.

[Wooldridge et al., 2005] Wooldridge, M., McBurney, P., and Parsons, S. (2005). On the metalogic of arguments. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 560–567, New York, NY, USA. Association for Computing Machinery.

[Xin et al., 2018] Xin, D., Ma, L., Liu, J., Macke, S., Song, S., and Parameswaran, A. (2018). Accelerating human-in-the-loop machine learning: Challenges and opportunities. In *Proceedings of the second workshop on data management for end-to-end machine learning*, pages 1–4.

[Xin et al., 2021] Xin, D., Wu, E. Y., Lee, D. J. L., Salehi, N., and Parameswaran, A. G. (2021). Whither automl? understanding the role of automation in machine learning workflows. In *CHI '21: CHI Conference on Human Factors in Computing Systems*, pages 83:1–83:16. ACM.

[Yaghini et al., 2021] Yaghini, M., Krause, A., and Heidari, H. (2021). A human-in-the-loop framework to construct context-aware mathematical notions of outcome fairness. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 1023–1033.