Arg2P: An argumentation framework for explainable intelligent systems

Arg2P: An argumentation framework for explainable intelligent systems*

Roberta Calegari[1][0000−0003−3794−2942], Andrea Omicini[2][0000−0002−6655−3869], Giuseppe Pisano[1][0000−0003−0230−8212], and Giovanni Sartor[1][0000−0003−2210−0398]

[1] Alma AI – Alma Mater Research Institute for Human-Centered Artificial Intelligence, Alma Mater Studiorum—Università di Bologna, Italy
[2] Dipartimento di Informatica – Scienza e Ingegneria (DISI), Alma Mater Studiorum—Università di Bologna, Italy

Abstract. In this paper we present the computational model of Arg2P, a logic-based argumentation framework for defeasible reasoning and agent conversation particularly suitable for explaining agent intelligent behaviours. The model is reified as the Arg2P technology, which is presented and discussed both from an architectural and a technological perspective so as to point out its potential in the engineering of intelligent systems. Finally, an illustrative application scenario is discussed in the domain of computable law for autonomous vehicles.

Keywords: Arg2P · intelligent systems engineering · explainable intelligent systems · logic-based technology · argumentation · defeasible reasoning · multi-agent systems

1 Introduction

Intelligent systems nowadays can be generally understood as socio-technical systems composed of human and artificial agents, computational and physical artefacts, as well as institutions and norms regulating interactions among heterogeneous components. The design of intelligent socio-technical systems calls for non-trivial social and organisational concepts and techniques, typically borrowed from the field of *agents* and *multi-agent systems* (MAS henceforth) [54]. În particular, *agreement technologies* [55] enable intelligent interaction among autonomous agents aimed at promoting cooperation and collaborative activities within intelligent systems—such as dialogue, negotiation, argumentation.

Given their their long-term connection with MAS [12, 49], *logic-based technologies* have a role to play in that context, in particular when dealing with interaction—both human-to-agent and agent-to-agent [52]. More specifically, logic-based *agreement technologies* can work as a general framework for *defeasible reasoning* and agent conversation, where *argumentation* plays a central role

* Roberta Calegari and Giovanni Sartor have been supported by the H2020 ERC Project "CompuLaw" (G.A. 833647). Andrea Omicini has been supported by the H2020 Project "AI4EU" (G.A. 825619).

[15]. Argumentation is particularly relevant in the legal context [16], especially when intelligent agents are required not just to agree upon some conclusion, but also to make their reasoning understandable to humans and other agents—to *explain* themselves [18].

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 [13]. This is why in this work we present the Arg2P *technology*[3] and its *computational model*, based on the formal model and its proof already discussed in [21].[4] In order to fit the most advanced application scenarios for intelligent systems – such as pervasive intelligent systems, or the Internet of Intelligent Things (IoIT) [3] – the Arg2P framework is specifically designed according to the definition of *micro-intelligence* [53, 11], 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.

Our work here builds over our previous contribution [56], where we discussed an early prototype of the Arg2P system. First of all, the Arg2P model is extended to integrate all the major features of the ASPIC$^+$ framework—since previous versions did not support strict knowledge and exploited a more limited set of preferences orderings. The implementation has been vastly improved—to cite just one specific improvement, the query-based mode of the engine has been completely re-designed for the sake of efficiency. Moreover, here we focus the discussion on the purposes of the engine, by giving a practical demonstration of its potentialities in a hypothetical application scenario.

In line with the micro-intelligence definition, the Arg2P framework is a general-purpose inference engine offering a number of diverse reasoning capabilities – not just defeasible reasoning – to be easily exploited for agent-based intelligent systems within a wide range of different applications scenarios, as discussed in [16]. Moreover, its modular architecture offers a neutral ground upon which different technologies can be integrated to improve the explainability of intelligent systems. For this reason, in this paper we choose as our running examples the computable law and legal aspects of autonomous vehicles, as this allows the relevant aspects of Arg2P to be highlighted in a hot-topic literature case study [29]. Clearly, a full discussion of the requirements for the specific application scenario is out of the scope of this work: so, far from being exhaustive, the discussion of the application scenario is only illustrative for the purposes of the engineering of explainable intelligent systems with argumentation.

The work is structured as follows. Section 2 overviews the available technologies in the argumentation field while introducing the motivations that led

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

[4] For formal accounts and model integration in MAS, which are out of the scope of this paper, please refer to [17, 19–21, 50, 16].

to the creation of the Arg2P engine. Section 3 discusses the formal foundations of the framework, while Section 4 focuses on the framework from an algorithmic perspective. Section 5 discusses the architecture and the engine interface; then, Section 6 reifies the model into a suitable argumentation language for the operational use of the tool, whereas Section 7 discuss technological details. Section 8 discusses our running example of computable law and legal aspects of autonomous vehicles [29] showcasing Arg2P potential in the engineering of intelligent systems. Section 9 provides for the final remarks.

2 Related works & motivations

Over the last twenty years, argumentation has become increasingly central in AI research. On the one side, the exploitation of elements from logic and formal deductive reasoning has provided a powerful foundation for computational models in complex AI settings. On the other side, many significant directions pursued in recent years have broadened the scope – and brought about new issues and challenges – of argumentation in AI. In the last decade we have witnessed a shift in emphasis within computational models of argumentation in AI that moves from formalisms rooted in classical deductive reasoning towards models that propose computational foundations to account for the dialogical aspects of the argumentation process. If, on the one hand, these features make argumentation particularly attractive to applications requiring distributed intelligence, autonomous components, and synchronous interaction, many open challenges remain to be faced, both theoretical – such as conceptual integrity and global model consistency – and pragmatical—such as computational efficiency.

Since a comprehensive survey of existing trends and works in the field is outside the scope of this paper, in the following we just recap the main computational models reified as available technologies that emerged over the last decades. Our purpose here is *(i)* to highlight the main features and the limitations of existing models and technologies when dealing with open and highly-distributed AI systems, and *(ii)* to position the Arg2P technology in the research context. For a more thorough discussion on the subject, we forward the interested reader to some recent works such as [6] and [25].

2.1 Abstract argumentation

The recent landscape of computational argumentation dates back to Dung's foundational work on *abstract argumentation theory* [35]. 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 [57] – exposing a direct implementation of labelling algorithms and aimed at the application in the context of MAS –, ASPARTIX [37] – based on an answer set programming (ASP) encoding of the argumentation problem –, ConArg [9] – a constraint-based framework –, and μ-toksia [51]—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)[5] 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 [7]. 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 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 [8] seems to be most appropriate [45]. 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 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 [40], Carneades [43], ABA [62], and ASPIC [50]. 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

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

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[6], also available on the Tweety library.[7] 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 [1, 2], but no mature implementation has emerged.

Carneades. The Carneades technology exposes an implementation of its model [44] offering both Dung's abstract reasoning and structural reasoning.[8] From a technological point of view, it is one of the best solutions that can be found. Written in Go [9] 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 [61], abagraph [31] and grapharg [32][10] – and to the abstract reductionist strand—such as ABAplus[11] and [47]. ABAplus [4] 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 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 [50, 41, 39]. The main implementations available

[6] `http://lidia.cs.uns.edu.ar/delp\_client/`
[7] `http://tweetyproject.org/`
[8] `http://carneades.fokus.fraunhofer.de/carneades/`
[9] `http://golang.org/`
[10] `http://robertcraven.org/proarg/`
[11] `http://www-abaplus.doc.ic.ac.uk/`

(Toast) [60] is based on an abstract reductionist approach exploiting Dung-O-Matic[12] as its base solver. There have been some attempts to perform structural reasoning in MAS exploiting ASPIC: among the others, notable examples are Argue tuProlog [10] and the ASPIC Argumentation Engine[13]; yet, the resulting technologies can be classified as just proofs of concept.

2.3 Arg2P motivations

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, [48] discusses how a direct declarative approach based on ASP can be developed, whereas [24] – implementing [22] – shows how argument-based entailment can be brought closer to human intuition, by proposing the use of formal discussion as a bridge technology.

In this lively panorama Arg2P lays its roots by sharing most of the motivations behind the aforementioned technologies, yet putting some more emphasis on the pervasive scope and interoperability requirements. In particular, the tool is designed according to the structured argumentation theories discussed above and provides well-defined semantics for defeasible reasoning and dialogue in open and dynamic systems.

Arg2P is largely inspired by the aforementioned technologies and methods: it ensures complete adherence to existing formal models and well-founded semantics and revisits some existing algorithms for the sake of efficiency. However, with respect to existing tools and technologies, Arg2P aims at stressing its exploitability in complex systems—in particular the pervasive ones. This is why it is designed around the concept of *micro-intelligence* [11]: there, the very foundation of the tool is not to act as a monolithic oracle, but as a lightweight tool for injecting intelligence and argumentation capabilities into the system when and where needed.

Moreover, argumentation and defeasible reasoning are not the only facets of the engine. In fact, in the very essence of the concept of micro-intelligence, there is the possibility of customising the inference method (easily interchangeable) according to the contingent application needs. Based on the notion of *ecosystem* of logic-based mechanisms as provided by its technical foundation (tuProlog/2P-Kt [28]), 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

[12] http://arg-tech.org/index.php/projects/dung-o-matic/
[13] http://webspace.science.uu.nl/~prakk101/aspic/

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, Arg2P 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 adhering to the *micro-intelligence* concept and based on current software engineering standard practices.

3 Model

This section summarises the main staples of the model. An extensive discussion on the theoretical model is out of the scope of this work, which focuses instead on the computational and technological aspects of the Arg2P engine. In the remainder of this section we present just the main aspects of the model, which are needed in order to understand the computational model and the technology described in the subsequent sections—for further details on the specific features of the model we defer readers to [19–21].

Arg2P is a modular rule-based argumentation system enabling representation, reasoning, and argumentation upon conditional norms, featuring obligations, prohibitions, and (strong or weak) permissions also according to burdens of persuasion constraints. The approach is based on common constructs of argument in computational models that lay their root in Dung's abstract argumentation [35] and structured argumentation [8].

Given an argumentation graph, the sets of arguments that are accepted or rejected – that is, those arguments that will survive or not to possible attacks – are computed according to semantics. For our purposes, here we leverage on *labelling semantics* [5]. Accordingly, we endorse {IN, OUT, UND}-labellings where each argument is associated with one label that is either IN, OUT, or UND, respectively meaning that the argument is accepted, rejected, or undecided. Language and semantics are extended for dealing with deontic extension and burdens of proof. In the following a recap of formal accounts of the model is provided.

3.1 Defeasible theories & arguments

Let a literal be an atomic proposition or its negation.

Notation 1 *For any literal ϕ, its complement is denoted by $\bar{\phi}$. That is, if ϕ is a proposition p, then $\bar{\phi} = \neg p$, while if ϕ is $\neg p$, then $\bar{\phi}$ is p.*

Literals are brought into relation through strict and defeasible rules.

Strict rules are rules in the classical sense: whenever the premises are indisputable (e.g., axioms), so is the conclusion.

Definition 1 (Strict rule). *A **strict rule** r has the form: $\rho : \phi_1, ..., \phi_n \rightarrow \psi$ with $0 \leq n$, and where*

- *ρ is the unique identifier for r, denoted by $\mathsf{N}(r)$;*
- *each $\phi_1, \ldots \phi_n, \psi$ is a literal;*
- *$\phi_1, \ldots \phi_n$ are denoted by Antecedent(r) and ψ by Consequent(r).*

A strict rule is an expression indicating that if $\phi_1, \ldots \phi_n$ hold, then without exception it holds that ψ. **Axioms** are defined via strict rules with no *Consequent*(r).

Defeasible rules are rules that can be defeated by contrary evidence.

Definition 2 (Defeasible rule). *A **defeasible rule** r takes the form: $\rho : \phi_1, ..., \phi_n, \sim \phi'_1, ..., \sim \phi'_m \Rightarrow \psi$ with $0 \leq n, m'$, and where*

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

Generally speaking, a set of *Rules* representing the knowledge is composed of the following disjoint subsets:

- Axm a set of axioms, defined via strict rules with no *Consequent*(r)
- NonAxm a set of non-axiom premises (i.e., the ordinary defeasible premises), defined via defeasible rules with no *Consequent*(r)
- $\mathsf{StrictRules}$ a set of strict rules
- $\mathsf{DefRules}$ a set of defeasible rules

i.e., $Rules = \{\mathsf{Axm} \cup \mathsf{NonAxm} \cup \mathsf{StrictRules} \cup \mathsf{DefRules}\}$.

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

Definition 3 (Superiority relation). *A **superiority relation** $\succ$ over a set of rules Rules is an antireflexive and antisymmetric binary relation over Rules, i.e., $\succ \subseteq Rules \times Rules$.*

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

Definition 4 (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 $\mathsf{DefRules} \cup \mathsf{NonAxm}$.*

Arguments are built from axioms and premises as well as from strict and defeasible rules. Given a defeasible theory, by chaining rules from the theory we can construct arguments, as specified in the following definition; cf. [50, 23, 63].

Definition 5 (Argument). *An **argument** A constructed from a defeasible theory $\langle Rules, \succ \rangle$ is a finite construct of the form:*

1. *$A : A_1, \ldots A_n \rightarrow_r \phi$ or*

2. $A: A_1, \ldots A_n \Rightarrow_r \phi$

with $0 \leq n$, *where*

- A *is the argument's unique identifier;*
- $A_1, \ldots, A_n$ *are arguments constructed from the defeasible theory* $\langle Rules, \succ \rangle$*;*
- *all the axioms and non-axioms of a defeasible theory used to build the argument are called premises and are denoted by* $\mathsf{Prem}(A)$*;*
- ϕ *is the* conclusion *of the argument, denoted by* $\mathsf{Conc}(A)$*;*
- $\mathsf{DefRules}(A)$ *is the set of defeasible rules exploited to build argument* A*;*
- $\mathsf{DefRules}(A) = \mathsf{DefRules}(A_1) \cup \ldots \cup \mathsf{DefRules}(A_n)$

and

1. $r : \mathsf{Conc}(A_1), \ldots, \mathsf{Conc}(A_n) \rightarrow \phi$ *is the top rule of* A*, denoted by* $\mathsf{TopRule}(A)$ *or*
2. $r : \mathsf{Conc}(A_1), \ldots, \mathsf{Conc}(A_n) \Rightarrow \phi$ *is the top rule of* A*, denoted by* $\mathsf{TopRule}(A)$*.*

Notation 2 *Given an argument* A *as in definition 5,* $\mathsf{Sub}(A)$ *denotes the* ***set of subarguments*** *of* A*, i.e.,* $\mathsf{Sub}(A) = \mathsf{Sub}(A_1) \cup \ldots \cup \mathsf{Sub}(A_n) \cup \{A\}$*.* $\mathsf{DirectSub}(A)$ *denotes the* ***direct subarguments*** *of* A*, i.e.,* $\mathsf{DirectSub}(A) = \{A_1, \ldots, A_n\}$*.*

Definition 6 (Argument properties). *An argument* A *is*

- strict *if* $\mathsf{DefRules}(A) = \emptyset$
- defeasible *if* $\mathsf{DefRules}(A) \neq \emptyset$
- firm *if* $\mathsf{Prem}(A) \subseteq \mathsf{Axm}$
- plausible *if* $\mathsf{Prem}(A) \cap \mathsf{NonAxm} \neq \emptyset$*.*

Preferences over arguments can be defined via last link or weakest link principle, as formalised in the following.

Definition 7 (Last defeasible rules). *Let* A *be an argument.*

- $\mathsf{LastDefRules}(A) = \emptyset$ *iff* $\mathsf{DefRules}(A) = \emptyset$
- *if* $A: A_1, \ldots A_n \Rightarrow \phi$*, then* $\mathsf{LastDefRules}(A) = \{\mathsf{Conc}(A_1), \ldots, \mathsf{Conc}(A_n) \Rightarrow \phi\}$ *else* $\mathsf{LastDefRules}(A) = \mathsf{LastDefRules}(A_1) \cup \ldots \mathsf{LastDefRules}(A_n)$*.*

Definition 8 (Last link principle). *A* ***preference relation*** $\succ$ *is a binary relation over a set of arguments* $\mathcal{A}$*: an argument* A *is preferred to argument* B*, denoted by* $A \succ B$*, iff*

- $\mathsf{LastDefRules}(A) \succ \mathsf{LastDefRules}(B)$*; or*
- $\mathsf{LastDefRules}(A) = \emptyset, \mathsf{LastDefRules}(B) = \emptyset$ *and* $\mathsf{Prem}(A) \succ \mathsf{Prem}(B)$*.*

Definition 9 (Weakest link principle). *A* ***preference relation*** $\succ$ *is a binary relation over a set of arguments* $\mathcal{A}$*: an argument* A *is preferred to argument* B*, denoted by* $A \succ B$*, iff*

- A *and* B *are strict and* $\mathsf{Prem}(A) \succ \mathsf{Prem}(B)$*; else*
- A *and* B *are firm and* $\mathsf{DefRules}(A) \succ \mathsf{DefRules}(B)$*; else*

– $\mathsf{Prem}(A) \succ \mathsf{Prem}(B)$ *and* $\mathsf{DefRules}(A) \succ \mathsf{DefRules}(B)$.

In the same way as ASPIC$^+$, Arg2P introduces two alternatives to be used in the above definitions to compare set of rules: namely, the *elitist* and *democratic* superiority relations.

Definition 10 (Elistist superiority relation). *Given two set of defeasible rules/premises A and B, $A \succ B$ iff $\exists X \in B$ s.t. $\forall Y \in A$ we have that $Y \succ X$.*

Definition 11 (Democratic superiority relation). *Given two set of defeasible rules/premises A and B, $A \succ B$ iff $\forall X \in B$, $\exists Y \in A$ s.t. $Y \succ X$.*

3.2 Defeat

Rebuts, undermining, and undercutting are defined as usual—see [50].

Definition 12 (Defeat). *An argument A defeats an argument B iff A undercuts, rebuts or undermines B, where:*

- *A undercuts B iff $\mathsf{Conc}(A) = \neg\mathsf{N}(r)$ for some $B' \in \mathsf{Sub}(B)$ such that B''s top rule r is defeasible;*
- *A rebuts B iff $\mathsf{Conc}(A) = \neg\phi$ for some $B' \in \mathsf{Sub}(B)$ of the form $B'_1, \ldots, B'_n \Rightarrow \phi$;*
- *A undermines B iff $\mathsf{Conc}(A) = \neg\phi$ for an ordinary premise ϕ of B.*

Defeat with burdens of persuasion The burden of persuasion allows a response on the acceptability of an argument to be obtained even in circumstances that would normally be uncertain. Generally speaking, the burden of persuasion specifies which party has to prove a statement to a specified degree (the standard of proof) on the penalty of losing on the issue. Whether this burden is met is determined in the final stage of a proceeding, after all evidence is presented. The burden of persuasion for a claim can be defined – under a logical perspective – as the task of ensuring that in the final stage of the proceeding there exists a justified argument for the claim.

Let us first 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 both on ϕ and $\overline{\phi}$.

Definition 13 (Burdens of persuasion). *Let $\mathsf{BurdPers}$, the set of **burdens of persuasion**, be a set of literals such that if $\phi \in \mathsf{BurdPers}$ then $\overline{\phi} \notin \mathsf{BurdPers}$. We say that an argument A is burdened with persuasion if $\mathsf{Conc}(A) \in \mathsf{BurdPers}$.*

We now consider possible collisions between arguments, i.e., those cases in which an argument A challenges an argument B: *(a)* by contradicting the conclusion of a B' subargument, or *(b)* by denying (the application of) the top rule of a B' subargument or by contradicting a weak negation in the body of the top rule of a B' subargument. Note that our notion of rebutting corresponds to the notion of successful rebutting in [58].

Definition 14 (bp-rebut). *Argument A* ***bp-rebuts*** *argument B iff $\exists B' \in Sub(B)$ such that $\mathsf{Conc}(A) = \overline{\mathsf{Conc}(B')}$ and*

1. *$\mathsf{Conc}(A) \notin \mathsf{BurdPers}$, and $B' \not\succ A$, or*
2. *$\mathsf{Conc}(A) \in \mathsf{BurdPers}$ and $A \succ B'$.*

According to definition 14.1, for an unburdened argument A to rebut B by contradicting the latter subargument B', it is sufficient that B' is non-superior to A. According to 14.2 for a burdened argument A to rebut B by contradicting B', it is necessary that A is superior to B'. Thus, burdens of persuasion supplement priorities in deciding conflicts between arguments having opposed conclusions. They dictate the outcome of those conflicts when priorities do not already determine which argument is to prevail: when two arguments contradict one another, the one burdened with persuasion will fail to bp-rebut the other, while the latter will succeed in bp-rebutting the first.

Undercutting is defined as both the case when the attacker excludes the application of the top rule of the attacked argument (by denying the rule's name) and the case when it contradicts a weakly-negated literal in the body of that rule.

Definition 15 (bp-undercut). *A* ***undercuts*** *B iff $\exists B' \in Sub(B)$ such that:*

1. *$\mathsf{Conc}(A) = \neg \mathsf{N}(r)$ and $\mathsf{TopRule}(B') = r$; or*
2. *$\mathsf{Conc}(A) = \phi$ and $\sim \phi \in Antecedent(\mathsf{TopRule}(B'))$*

Finally, we have the notions of bp-defeat and strict bp-defeat that are defined on the basis of bp-rebutting and undercutting. As one can see from the definition below, the difference w.r.t. the usual notion of defeat pertains to bp-defeat.

Definition 16 (bp-defeat).

1. *A* ***bp-defeats*** *B iff A bp-rebuts B or A undercuts B*
2. *A* ***strictly-bp-defeats*** *B iff A bp-defeats B and B does not bp-defeats A .*

3.3 Argumentation graphs, labelling & bp-labelling

In an argumentation graph, arguments are connected according to the defeat relation.

Definition 17 (Argumentation graph). *An* ***argumentation graph*** *constructed from a defeasible theory T is a tuple $\langle \mathcal{A}, \rightsquigarrow \rangle$, where $\mathcal{A}$ is the set of all arguments constructed from T, and $\rightsquigarrow$ is* defeat *relation over $\mathcal{A}$.*

Notation 3 *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 18 (Labelling). *Let G be an argumentation graph. An $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$-**labelling** L of G is a total function $\mathcal{A}_G \to \{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$.*

Notation 4 *Given a labelling L, we write $\mathsf{IN}(L)$ for $\{A|L(A) = \mathsf{IN}\}$, $\mathsf{OUT}(L)$ for $\{A|L(A) = \mathsf{OUT}\}$ and $\mathsf{UND}(L)$ for $\{A|L(A) = \mathsf{UND}\}$.*

Grounded and complete semantics are defined as usual c.f. [35]:

Definition 19 (Complete Labelling). *Let G be an argumentation graph. A complete labelling of G is a $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$-**labelling** L where every IN-labelled argument is legally IN, every OUT-labelled argument is legally OUT and every UND-labelled argument is legally UND, i.e., it holds that:*

1. *an argument A is labelled IN iff all its attackers are labelled OUT, and*
2. *an argument A is labelled OUT iff it has at least one attacker that is labelled IN.*

Definition 20 (Grounded Labelling). *Let G be an argumentation graph. The grounded labelling of G is a complete $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$-**labelling** L where $\mathsf{IN}(L)$ is minimal.*

We now specify the notion of **bp-labelling**, namely, a labelling which takes into account a set of burden of persuasion BurdPers.

Definition 21 (bp-labelling). *A **bp-labelling** of an argumentation graph G, relative to a set of burdens of persuasion BurdPers, is a $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$-labelling s.t. $\forall A \in \mathcal{A}_G$ with $\mathsf{Conc}(A) = \phi$*

1. *$A \in \mathsf{IN}(L)$ iff $\forall B \in \mathcal{A}_G$ such that B bp-defeats $A : B \in \mathsf{OUT}(L)$*
2. *$A \in \mathsf{OUT}(L)$ iff*
   (a) *$\phi \in \mathsf{BurdPers}$ and $\exists\ B \in \mathcal{A}_G$ such that*
      - *B bp-defeats A and*
      - *$B \in \mathsf{IN}(L)$ or $B \in \mathsf{UND}(L)$*

   (b) *$\phi \notin \mathsf{BurdPers}$ and $\exists\ B \in \mathcal{A}_G$ such that*
      - *B bp-defeats A and*
      - *$B \in \mathsf{IN}(L)$*
3. *$A \in \mathsf{UND}(L)$ otherwise.*

Burdens of persuasion affect conditions for rejection, as specified in Definition 21(2)(a). The rejection (the OUT labelling) of an argument burdened of persuasion may be determined by any counterargument B that is accepted (IN) or also is uncertain (UND). On the contrary, as specified in 21(2)(b) the rejection of an argument that is not burdened with persuasion requires a defeating counterargument B that is IN. Note that the semantic just described does not always deliver a single labelling. Multiple labelling may exist when arguments rebut each other, none of them being burdened with persuasion. If one of these arguments is labelled IN the other is labelled OUT and vice versa. In order to address that, we focus on IN-minimal labelling, i.e., on the labelling where both such arguments are labelled UND. Let us call such a labelling a *grounded bp-labelling*.

Definition 22 (Grounded bp-labelling). *A **bp-labelling** L of an argumentation graph G is a **grounded bp-labelling** iff $\mathsf{UND}(L)$ is maximal.*

3.4 Deontic rules and conflict

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 [59], 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 23 (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 equivalently 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 Fig. 1. Detailed formal accounts of the adopted deontic extensions are discussed

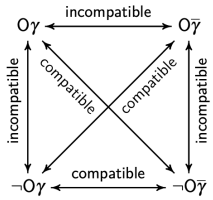

Fig. 1. Deontic square of compatibility relation

in [59]—our model fully adheres to that semantics.

4 Computational model & algorithms

The main and distinguishing aspect 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 aforementioned AI pervasive scenarios. Given the state of the art advancement in the computational model of the query mode operation, in the following, the discussion will focus only on that latter topic, first facing complexity issues and then detailing the steps of the algorithm.

4.1 Computational complexity

In the graph-based modality, computation is 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, at the moment, the engine allows for different semantics to be computed, according to the model defined in Section 3. Their complexity is a well-known problem in the literature – grounded semantic is the only one having polynomial complexity [46] –, however, we have not yet explored which algorithmic measure should be adopted in order to improve the computation. Up to now, we focus on the improvement of the query-based mode.

To ensure efficiency in this second operation mode – currently available only for grounded semantic – we exploit an algorithm inspired to the one introduced in DeLP [40] and discussed in the following. The structured reasoning algorithm avoids the entire argumentation graph construction for the evaluation of 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 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.

4.2 Structured reasoning algorithm

The algorithm used to evaluate a single claim (or query) is inspired by the dialectical trees from DeLP [40]. Starting from the given claim, arguments $A_1, ..., A_n$ sustaining that claim are constructed and evaluated. Then, starting from A_1, conflicting arguments – built by need – are recursively inspected to determine their admissibility.

Listing 1.1. Structured evaluation algorithm for grounded semantic

```
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
```

Listing 1.1 shows conditions under which the argument A_1 is evaluated. There, function $Attacker(A_1)$ returns a conflicting argument based on the notion of defeat defined in Subsection 3.2. The evaluated conditions are three—note that the order is important to guarantee the soundness of the algorithm:

1. if a conflicting argument labelled as IN exists, then A_1 is OUT;
2. if a cycle in the route from the root to the leaves ($Chain$) exists, then A_1 argument is UND;
3. if a conflicting argument labelled as UND exists, then also the A_1 argument is UND.

The second condition behaviour is in accordance with Dung's grounded semantic, which is why this algorithm only works for the grounded semantics. The structured evaluation under different semantics will be explored in future works.

If none of the above conditions is met then the A_1 argument is labelled IN. 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.

5 Architecture & API

The engine is designed in a fully-modular way, as depicted in Fig. 2. 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. In the following, we describe each component in detail.

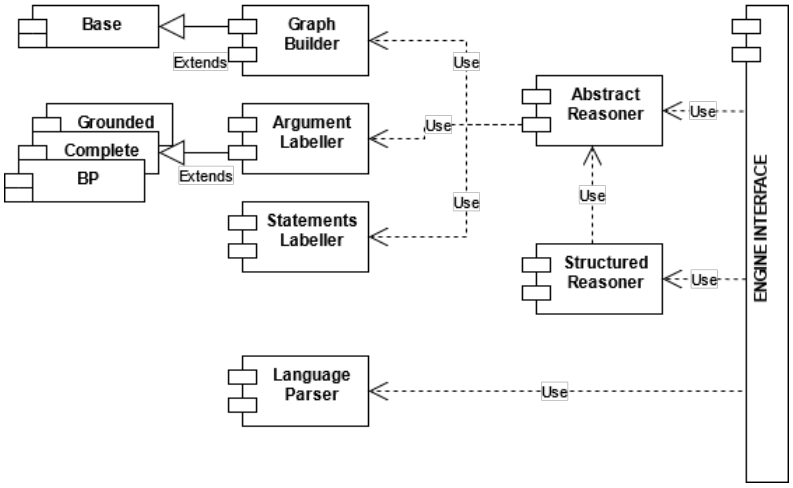

Fig. 2. Arg2P architecture

5.1 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, INArg, OUTArg, and UNDArg 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. According to the description provided in Section 3:

- `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 [5, 20]); 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`; the flag reflects the propagation strategies of definition 8 and definition 9, respectively;
- `orderingComparator(MODE)` sets the superiority relation to exploit for argument ranking: implemented strategies are `democrat` and `elitist`, both presented in [50], and `normal` from [36];
- `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 [23] 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 [23]; 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;

5.2 Core modules

In addition to the *Engine Interface* module, other modules depicted in Fig. 2 bear different responsibilities. In particular, they can be. split into three distinct groups of modules according to their purpose.

First, we have the group of the support modules, bearing transversal responsibilities w.r.t. the entire framework. The *Language parser* is currently the sole module belonging to this group: its role is to convert the rules from the argumentation language of the framework (Section 6) 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.

Then, the second group – the actual 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.

6 Language

The engine adopts an ASPIC$^+$-like syntax – introduced in [50] – and exposes all the main features of this framework. With respect to the original notation, the argumentation language has been extended with a specific notation for dealing with deontic operators and burdens of persuasion constraints.

In the following the language predicates reifying the above-described model are discussed.

6.1 Rules & premises

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

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

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

$$label \ : \ -> conclusion.$$
$$(label : [] \ -> conclusion.) \text{ (this form is also permitted)}$$

Defeasible rules are expressed by the => operator:

$$label : premise_1, \ldots, premise_n => conclusion.$$

Accordingly, defeasible premises take the form:

$$label : \ => conclusion.$$

Note that in the case of defeasible rules the rule

$$label : [] => conclusion.$$

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.

6.2 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:

- $-term$, 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:

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

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:

$$r_1 : look(Object, Colour) => colour(Object, Colour).$$
$$r_2 : illuminated(Object, Colour) => undercut(r_1).$$

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.

6.3 Deontic expressions

The formalism includes permissions, prohibitions, and obligations; the corresponding logic is captured by defeasible rule schemata, modelling basic deontic inference. In the language 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 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).

6.4 Superiority relation & burden of persuasion

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 $ruleName_2$.

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.

7 Technology

The Arg2P technology engine is built on the top of the tuProlog/2P-Kt engine [26, 33, 27] and provides a lightweight implementation of an ASPIC$^+$-like system for structured argumentation. The engine is available as a standalone application (Arg2P Java IDE), and as a 2P-Kt library (Kotlin library) [27, 28][14]. All the release are available on the Arg2P GitHub repository.[15]

The technological aspects of Arg2P are of particular relevance for understanding how Arg2P differs w.r.t. others available technologies. 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 [3] – we devote our attention to computational logic as

[14] `https://github.com/tuProlog/2p-kt`
[15] `https://github.com/tuProlog/arg2p-kt/releases`

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 Arg2P on the top of tuProlog—and in particular on 2P-Kt, a reboot of the tuProlog project offering a general, extensible, and interoperable ecosystem for LP and symbolic AI [27].

Leveraging on 2P-Kt – and thanks to its architecture (detailed in Section 5) –, the Arg2P 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.[16] 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 Arg2P framework natively supports interoperability with JVM, JavaScript and Android platforms.

Modularity & customisation. The entire Arg2P framework is a collection of tuProlog compatible libraries—thus enforcing system modularity, as discussed in Section 5. 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 Arg2P engine makes it possible to exploit diverse programming paradigms and technologies for any system component. In the current implementation, Arg2P 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.

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

Light-weightness & generality. In contrast to the current trend of building highly-specialised systems, the Arg2P framework places itself as a highly-general tool to be injected in most disparate environments for the most different application purposes. As shown in [12], the MAS community is gazing for interoperable and general-purpose logic-based technologies: there, the Arg2P 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 Arg2P 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 [3].

8 Application scenarios & discussion

The Arg2P framework has been designed to satisfy the needs of applications where distributed intelligence, autonomy, and interpretability are key requirements. In the following, 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. The complete guide of the examples, containing the corresponding theories and the detailed instructions for their execution within the framework can be found on the GitHub repository with the examples.[17]

The examples do not include a comprehensive analysis of the selected application domain: instead, they mean to underline the available features of the Arg2P engine and its ability to adapt to different contexts. Accordingly, we briefly recap those notions the reader needs to understand the examples: for a complete overview of the application scenario (computable law of autonomous vehicles), its requirement analysis, and design, please refer to [29] and related outcomes. In addition, readers can refer to some well-known works in literature, such as [42]. However, we also explicitly mention the requirements in the discussion of the examples whenever possible and useful.

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.

8.1 Example 1: Autonomous cars

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

[17] `https://github.com/tuProlog/JLC-SpecialIssue2021-arg2p-examples`

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 1.2 encodes the rules in the Arg2P system, whereas Listing 1.3 encodes the corresponding arguments.

Listing 1.2. 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 1.3. Arguments from Listing 1.2

```
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  ⟹  o(stop(ambulance))
A10 : A4,A7,r1  ⟹  o(stop(car))
A11 : A5,A8,r2  ⟹  p(-stop(pedestrian))
A12 : A3,A1,A0,A2,r3  ⟹  emergency(ambulance)
A13 : A5,A12,A8,r5  ⟹  o(stop(pedestrian))
A14 : A3,A12,A6,r4  ⟹  p(-stop(ambulance))
```

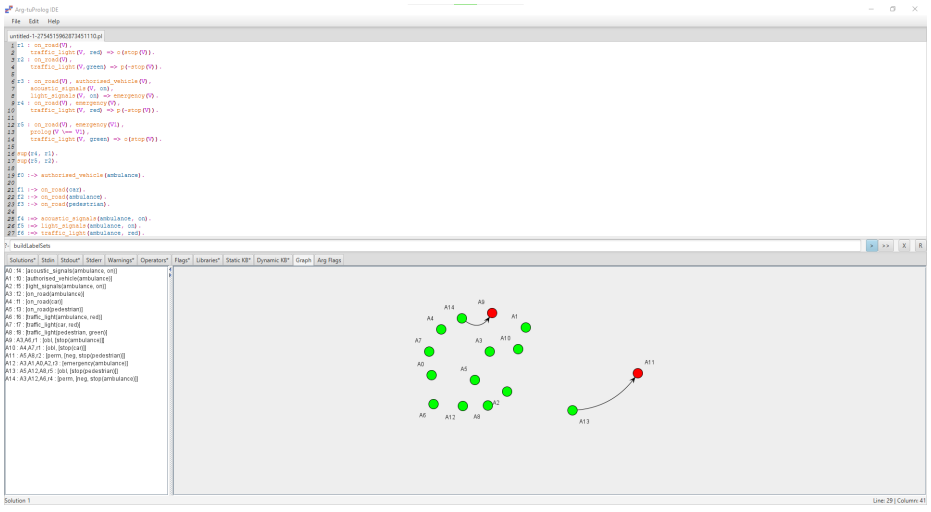

Fig. 3. Example 1 grounded argumentation graph in Arg2P IDE

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.[18] Finally, 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 (Fig. 3). Essential to this outcome is the emergency state of the ambulance (A_{12}): if it were not possible to prove the 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 1.4. 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)).
```

[18] 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 would lead to emergency vehicles having the obligation to stop at their own passage.

Listing 1.5. Arguments from Listing 1.2 and 1.4

```
A0 : f4  ⟹  acoustic_signals(ambulance, on)
A1 : f0  ⟹  authorised_vehicle(ambulance)
A2 : f16  ⟹  claim(chris, low_speed(ambulance))
A3 : f17  ⟹  claim(john, high_speed(ambulance))
A4 : f13  ⟹  harms(lisa, pino)
A5 : f5  ⟹  light_signals(ambulance, on)
A6 : f11  ⟹  -stop(ambulance)
A7 : f12  ⟹  -stop(pedestrian)
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  ⟹  -high_speed(ambulance)
A20 : A8,A11,r1  ⟹  o(stop(ambulance))
A21 : A9,A12,r1  ⟹  o(stop(car))
A22 : A10,A13,r2  ⟹  p(-stop(pedestrian))
A23 : A8,A1,A0,A5,r3  ⟹  emergency(ambulance)
A24 : A7,A22,r6  ⟹  legitimate_cross(pedestrian)
A25 : A18,A14,r11  ⟹  -careful(lisa)
A26 : A6,A20,r7  ⟹  -legitimate_cross(ambulance)
A27 : A26,A14,r10  ⟹  -careful(lisa)
A28 : A4,A25,r8  ⟹  responsible(lisa)
A29 : A10,A23,A13,r5  ⟹  o(stop(pedestrian))
A30 : A8,A23,A11,r4  ⟹  p(-stop(ambulance))
A31 : A4,A27,r8  ⟹  responsible(lisa)
A32 : A6,A30,r6  ⟹  legitimate_cross(ambulance)
A33 : A7,A29,r7  ⟹  -legitimate_cross(pedestrian)
A34 : A33,A15,r9  ⟹  responsible(pino)
A35 : A4,A34,r9  ⟹  responsible(pino)
A36 : A32,A19,A14,r12  ⟹  careful(lisa)
```

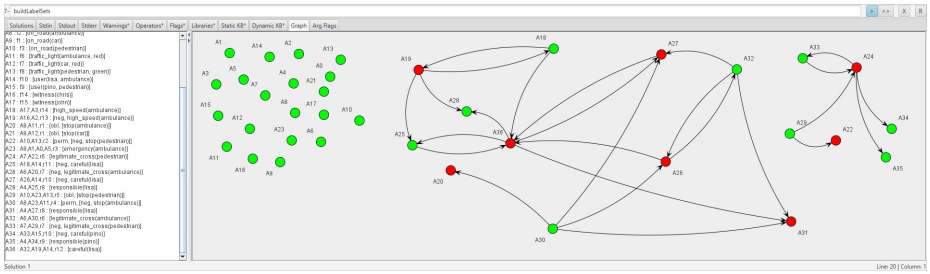

Fig. 4. Example 2 bp labelling in Arg2P IDE

8.2 Example 2: Autonomous cars & legal reasoning

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. 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 1.4 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 (Fig. 4) 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 1.6. 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)).
```

Listing 1.7. Arguments from Listing 1.2, 1.4 and 1.6

```
A0 : f4  ⟹  acoustic_signals(ambulance, on)
A1 : f0  ⟹  authorised_vehicle(ambulance)
A2 : f16  ⟹  claim(chris, low_speed(ambulance))
A3 : f17  ⟹  claim(john, high_speed(ambulance))
A4 : f25  ⟹  claim(mike, test_ok(ambulance))
A5 : f22  ⟹  claim(paul, defect(ambulance))
A6 : f13  ⟹  harms(lisa, pino)
A7 : f5  ⟹  light_signals(ambulance, on)
A8 : f19  ⟹  manufacturer(demers, ambulance)
A9 : f21  ⟹  mechanic(paul)
A10 : f11  ⟹  -stop(ambulance)
A11 : f12  ⟹  -stop(pedestrian)
A12 : f23  ⟹  new(ambulance)
A13 : f2  ⟹  on_road(ambulance)
A14 : f1  ⟹  on_road(car)
A15 : f3  ⟹  on_road(pedestrian)
A16 : f24  ⟹  production_manager(mike)
A17 : f6  ⟹  traffic_light(ambulance, red)
A18 : f7  ⟹  traffic_light(car, red)
A19 : f8  ⟹  traffic_light(pedestrian, green)
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  ⟹  -high_speed(ambulance)
A28 : A9,A5,r17  ⟹  -working(ambulance)
A29 : A13,A17,r1  ⟹  o(stop(ambulance))
A30 : A14,A18,r1  ⟹  o(stop(car))
A31 : A15,A19,r2  ⟹  p(-stop(pedestrian))
A32 : A13,A1,A0,A7,r3  ⟹  emergency(ambulance)
A33 : A11,A31,r6  ⟹  legitimate_cross(pedestrian)
A34 : A26,A21,r11  ⟹  -careful(lisa)
A35 : A28,A12,r18  ⟹  -defect_free(ambulance)
A36 : A10,A29,r7  ⟹  -legitimate_cross(ambulance)
A37 : A20,A21,A28,r16  ⟹  careful(lisa)
A38 : A36,A21,r10  ⟹  -careful(lisa)
A39 : A6,A34,r8  ⟹  responsible(lisa)
A40 : A15,A32,A19,r5  ⟹  o(stop(pedestrian))
A41 : A13,A32,A17,r4  ⟹  p(-stop(ambulance))
A42 : A6,A38,r8  ⟹  responsible(lisa)
A43 : A10,A41,r6  ⟹  legitimate_cross(ambulance)
A44 : A11,A40,r7  ⟹  -legitimate_cross(pedestrian)
A45 : A44,A22,r10  ⟹  -careful(pino)
A46 : A6,A21,A28,A8,A35,r15  ⟹  responsible(demers)
A47 : A6,A45,r9  ⟹  responsible(pino)
A48 : A43,A27,A21,r12  ⟹  careful(lisa)
```

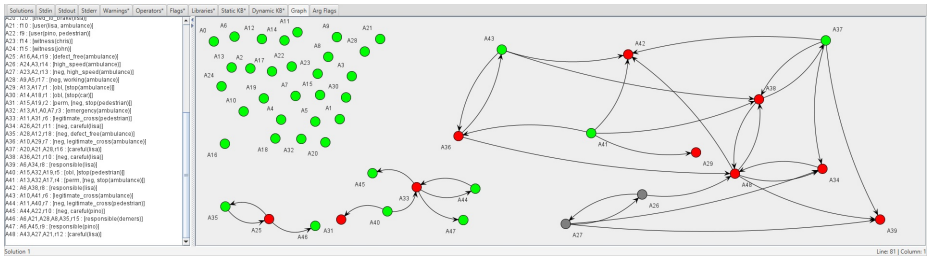

Fig. 5. Example 3 bp labelling in Arg2P IDE

8.3 Example 3: Autonomous cars, 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 1.6 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.

8.4 Examples discussion

As it emerges from the above-discussed examples, Arg2P is a *non-monotonic reasoner*, capable of improving the overall *system transparency* while enabling system engineers to deeply *customise* the behaviour of the entire engine tailoring their specific application context. In the remainder of this section, we discuss the Arg2P key features as emerged by the above example in relation to distributed pervasive multi-agent systems requirements.

System transparency & explainability. System transparency is easily obtained thanks to the intrinsic interpretability of argumentation models (leveraging on rule base approaches), thus ensuring key system features such as observability, interpretability, explicability, accountability, and trustability. An example of this property is highlighted in the first example presented in Subsection 8.1. Observing the knowledge encoding, as well as the output of the system in terms of decisions, it is immediately clear that both coding and decisions are highly interpretable and human-readable, making it easy for any observer to understand the output of the system. Let us suppose that an autonomous vehicle is capable of acquiring data from the surrounding environment (via car sensors) and can consequently update the knowledge base of its control system. The use of argumentation techniques would lead to a completely understandable and explainable

model of vehicle behaviours. Quite obviously, the use of this technology in that complex scenario would first require a neat upgrade in terms of overall system efficiency. To this end, as well as for knowledge encoding issues, advancements in the fields of neuro-symbolic computing and text mining could be exploited [38, 34].

The transparency of the system, along with the ability to argue about a decision, allow trustworthiness to be reached. This is a crucial aspect of ethical autonomous systems, and can be obtained by exploiting the argumentation intrinsic justification mechanisms: the acceptance (or rejection) of arguments and their premises in the framework provides an explanation for why an action was selected (or not).

Customisation. Customisation, under a technology perspective, often means easiness of integration with other techniques—for instance, already existing AI approaches. The second example (Subsection 8.2) underlies the advantages of the Arg2P modular architecture and the easiness of customising its inference capabilities by adding for instance a new semantic, like the burden of persuasion one. New semantics and customised reasoning mechanisms can be easily added by just adding the corresponding module that better fits the application scenarios. Accordingly, new modules where the reasoning process is integrated with other domain-specific techniques can be easily envisioned. The example focuses on Arg2P customisability in the computable law domain by introducing the burden of persuasion semantics in compliance with its related definitions. In particular, the example shows how in certain circumstances, in compliance with the law, responsibility must be established not only on the basis of what happened but also by demonstrating that safety rules have been respected. In the case at hand, purely illustrative, being careful is stated as a safety rule and is strictly correlated to the speed of the vehicle. Obviously, the scenario could be further complicated, but the key point here is to highlight the ease of grafting new features—in terms of predicates, argumentative semantics, or inference capabilities. The example examines the inclusion and use of the new burden semantic in order to give an output suitable to the application desiderata: the system, conscious of the legal context (*situateness* property) is given the proper tools to act accordingly. Of course, the same could happen for other specific needs: for example, [14] presents an extension enabling in Arg2P the notion of *jurisdiction*—one of the other properties selected in [42].

Non-monotonicity & defeasibility. The whole application scenario, and in particular the third example (Subsection 8.3), shows how non-monotonicity – a key feature of argumentation – fits perfectly the reasoning schemes required by real-world scenarios. In fact, knowledge is usually unstable, mutable, and contradicting, thus forcing intelligent agents to modify their goals, plans, and beliefs. The selected scenario highlights all the benefits of an easy-injectable technology – such as Arg2P – capable of delivering customisable reasoning capabilities in compliance with legal and normative standards.

Moreover, the case study shows how Arg2P meets many of the requirements discussed in [42] w.r.t. the treatment of the legal knowledge—i.e. *(i)* rules form and strict semantic, *(ii)* support for knowledge defeasibility through conflicts and exclusionary rules, *(iii)* contraposition (i.e., negative conclusion does not affect premises in defeasible rules), *(iv)* attribution of a value over rules through preferences, *(v)* deontic effects.

9 Conclusion & future work

In this work we present the Arg2P framework, focussing in particular on its computational model and on the technology. The work shows the effectiveness of Arg2P to *(i)* deal with inconsistent information—thus enabling defeasible reasoning; *(ii)* integrate legal aspects and custom behaviours—e.g., the possibility to explicit the burden of persuasion over terms; *(iii)* provide an easy and straightforward way to manipulate and interact with the engine.

One of the main strengths of the tool lays in its architecture: modularity, on one side, and full integration with LP, on the other side, make Arg2P highly suitable for intelligent pervasive systems. Moreover, its integration with LP – enhanced with argumentation capabilities – makes it easier for Arg2P to meet interpretability, understandability, and explainability requirements.

Even though the prototype technology presented in this work is already mature enough for tests and experiments, there is still a huge potential for improvements, in particular in terms of efficiency and further research challenges.

Efficiency issues. The main limit of the prototype is related to the query-based mode. In fact, the proposed algorithm only supports grounded semantics. Efficient solutions should also be designed for the other possible semantics. The second issue concerns the building mechanism for the argumentation graph. In order to derive the argumentation trees from facts and rules, an exhaustive search on the knowledge base is performed. Consequently, the most expensive operators affecting such a search should be implemented more efficiently—for instance, leveraging the tuProlog capabilities of integrating rules written directly in Java/Kotlin, and implementing an interpreter instead of the meta-interpreter proposed in the prototype.

Research challenge & issues. From the point of view of the research, Arg2P represents, in our vision, a fundamental brick for distributing symbolic intelligence in intelligent systems in compliance with the concept of micro-intelligence. Enabling argumentation and agreement capabilities should enhance system actors with properties like understandability and explainability – since the actors can argue over their decisions – but also normative enforcement—since actors can act in compliance with the law, and violations can be timely observed. Under this perspective, many challenges open up. For example, an open point is how to manage the resolution of conflicts among agents. A single "arbitrator" to manage the whole argumentation process and force the other agents' behaviour would

certainly be impractical in distributed contexts, since it would soon become a bottleneck. One could perhaps think about "area arbitrator" responsible for a certain space and for a certain period of time – also considering the dynamism of knowledge and its change over time – but the best solution still has to be designed and tested. Furthermore, in modern pervasive contexts, symbolic techniques need to be suitably integrated with sub-symbolic algorithms in order to exploit synergies and benefits of each approach in a fruitful way. For instance, a promising synergy point that is gaining attention in the recent literature is the use of graph neural networks (GAN) to predict the admissibility of arguments [30]. Indeed, as already acknowledged in the literature, argumentation is computationally complex (usually NP for computations other than grounded). In computationally-constrained scenarios, it is therefore directly not applicable, or at least it could be suitably integrated with other approaches, balancing advantages and disadvantages. For instance, GANs could be exploited to assess arguments' admissibility, losing some reliability in terms of correctness and for sure some transparency, but gaining efficiency. In general, the advancements in the fields of *neuro-symbolic computing* and *text mining* [38, 34] could help in mitigating efficiency and knowledge encoding issues, strictly related to the logic-based approach.

Overall, its high interoperability and modular architecture make Arg2P a useful technology addressing most of the aforementioned issues.

References

1. Alsinet, T., Béjar, R., Godo, L.: A characterization of collective conflict for defeasible argumentation. In: Computational Models of Argument. Frontiers in Artificial Intelligence and Applications, vol. 216, pp. 27–38. IOS Press (2010). https://doi.org/10.3233/978-1-60750-619-5-27
2. Alsinet, T., Béjar, R., Godo, L., Guitart, F.: Using answer set programming for an scalable implementation of defeasible argumentation. In: IEEE 24th International Conference on Tools with Artificial Intelligence. pp. 1016–1021. IEEE Computer Society (2012). https://doi.org/10.1109/ICTAI.2012.171
3. Arsénio, A., Serra, H., Francisco, R., Nabais, F., Andrade, J., Serrano, E.: Internet of Intelligent Things: Bringing artificial intelligence into things and communication networks. In: Inter-cooperative Collective Intelligence: Techniques and Applications, Studies in Computational Intelligence, vol. 495, pp. 1–37. Springer (2014). https://doi.org/10.1007/978-3-642-35016-0_1
4. Bao, Z., Cyras, K., Toni, F.: ABAplus: Attack reversal in abstract and structured argumentation with preferences. In: PRIMA 2017: Principles and Practice of Multi-Agent Systems. Lecture Notes in Computer Science, vol. 10621, pp. 420–437. Springer (2017). https://doi.org/10.1007/978-3-319-69131-2
5. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. The Knowledge Engineering Review **26**(4), 365–410 (2011). https://doi.org/10.1017/S0269888911000166
6. Baroni, P., Toni, F., Verheij, B.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games: 25 years later. Argument & Computation **11**(1-2), 1–14 (2020). https://doi.org/10.3233/AAC-200901

7. Bench-Capon, T., Coenen, F., Orton, P.: Argument-based explanation of the British nationality act as a logic program. Information and Communications Technology Law **2**(1), 53–66 (1993). https://doi.org/10.1080/13600834.1993.9965668
8. Besnard, P., García, A.J., Hunter, A., Modgil, S., Prakken, H., Simari, G.R., Toni, F.: Introduction to structured argumentation. Argument & Computation **5**(1), 1–4 (2014). https://doi.org/10.1080/19462166.2013.869764
9. Bistarelli, S., Santini, F.: Conarg: A constraint-based computational framework for argumentation systems. In: IEEE 23rd International Conference on Tools with Artificial Intelligence. pp. 605–612. Institute of Electrical and Electronics Engineers (2011). https://doi.org/10.1109/ICTAI.2011.96
10. Bryant, D., Krause, P.J., Vreeswijk, G.: Argue tuProlog: A lightweight argumentation engine for agent applications. In: Computational Models of Argument. Frontiers in Artificial Intelligence and Applications, vol. 144, pp. 27–32. IOS Press (2006), http://www.booksonline.iospress.nl/Content/View.aspx?piid=1922
11. Calegari, R.: Micro-Intelligence for the IoT: Logic-based Models and Technologies. Ph.D. thesis, Alma Mater Studiorum—Università di Bologna, Bologna, Italy (Apr 2018). https://doi.org/10.6092/unibo/amsdottorato/8521.
12. Calegari, R., Ciatto, G., Mascardi, V., Omicini, A.: Logic-based technologies for multi-agent systems: A systematic literature review. Autonomous Agents and Multi-Agent Systems **35**(1), 1:1–1:67 (2021). https://doi.org/10.1007/s10458-020-09478-3
13. Calegari, R., Contissa, G., Lagioia, F., Omicini, A., Sartor, G.: Defeasible systems in legal reasoning: A comparative assessment. In: Araszkiewicz, M., Rodríguez-Doncel, V. (eds.) Legal Knowledge and Information Systems. JURIX 2019: The Thirty-second Annual Conference, Frontiers in Artificial Intelligence and Applications, vol. 322, pp. 169–174. IOS Press (11-13 Dec 2019). https://doi.org/10.3233/FAIA190320
14. Calegari, R., Contissa, G., Pisano, G., Sartor, G., Sartor, G.: Arg-tuProlog: a modular logic argumentation tool for PIL. In: Villata, S., Harašta, J., Křemen, P. (eds.) Legal Knowledge and Information Systems. JURIX 2020: The Thirty-third Annual Conference. Frontiers in Artificial Intelligence and Applications, vol. 334, pp. 265–268 (9-11 Dec 2020). https://doi.org/10.3233/FAIA200880
15. Calegari, R., Omicini, A., Sartor, G.: Argumentation and logic programming for explainable and ethical AI. In: Musto, C., Magazzeni, D., Ruggieri, S., Semeraro, G. (eds.) XAI.it 2020 – Italian Workshop on Explainable Artificial Intelligence 2020. CEUR Workshop Proceedings, vol. 2742, pp. 55–68. Sun SITE Central Europe, RWTH Aachen University (Nov 2020), http://ceur-ws.org/Vol-2742/paper5.pdf
16. Calegari, R., Omicini, A., Sartor, G.: Computable law as argumentation-based MAS. In: Calegari, R., Ciatto, G., Denti, E., Omicini, A., Sartor, G. (eds.) WOA 2020 – 21st Workshop "From Objects to Agents". CEUR Workshop Proceedings, vol. 2706, pp. 54–68. Sun SITE Central Europe, RWTH Aachen University, Aachen, Germany (Oct 2020), http://ceur-ws.org/Vol-2706/paper10.pdf
17. Calegari, R., Omicini, A., Sartor, G.: Burdens of persuasion and standards of proof in structured argumentation. In: Baroni, P., Benzmüller, C., Wáng, Y.N. (eds.) Logic and Argumentation. pp. 40–59. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-89391-0_3
18. Calegari, R., Omicini, A., Sartor, G.: Explainable and ethical AI: A perspective on argumentation and logic programming. In: Baldoni, M., Bandini, S. (eds.) AIxIA 2020 – Advances in Artificial Intelligence, Lecture Notes in Computer Science,

vol. 12414, pp. 19–36. Springer Nature (2021). https://doi.org/10.1007/978-3-030-73065-9_2
19. Calegari, R., Riveret, R., Sartor, G.: The burden of persuasion in structured argumentation. In: Maranhão, J., Wyner, A.Z. (eds.) 17th International Conference on Artificial Intelligence and Law (ICAIL'21). pp. 180–184. ICAIL'21, ACM (Jun 2021). https://doi.org/10.1145/3462757.3466078
20. Calegari, R., Sartor, G.: 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., Riguzzi, F. (eds.) 36th International Conference on Logic Programming (ICLP 2020). Electronic Proceedings in Theoretical Computer Science, vol. 325, pp. 151–163. Open Publishing Association (Sep 2020). https://doi.org/10.4204/EPTCS.325.21
21. Calegari, R., Sartor, G.: A model for the burden of persuasion in argumentation. In: Villata, S., Harašta, J., Křemen, P. (eds.) Legal Knowledge and Information Systems. JURIX 2020: The Thirty-third Annual Conference. Frontiers in Artificial Intelligence and Applications, vol. 334, pp. 13–22 (9-11 Dec 2020). https://doi.org/10.3233/FAIA200845
22. Caminada, M.: A discussion game for grounded semantics. In: Theory and Applications of Formal Argumentation. Lecture Notes in Computer Science, vol. 9524, pp. 59–73. Springer (2015). https://doi.org/10.1007/978-3-319-28460-6
23. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence **171**(5–6), 286–310 (2007). https://doi.org/10.1016/j.artint.2007.02.003
24. Caminada, M., Uebis, S.: An implementation of argument-based discussion using ASPIC-. In: Computational Models of Argument. Frontiers in Artificial Intelligence and Applications, vol. 326, pp. 455–456. IOS Press (2020). https://doi.org/10.3233/FAIA200531
25. Cerutti, F., Gaggl, S.A., Thimm, M., Wallner, J.P.: Foundations of implementations for formal argumentation. Journal of Applied Logics—IfCoLog Journal of Logics and their Applications **4**(8), 2623–2705 (2017), `http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf`
26. Ciatto, G., Calegari, R., Omicini, A.: 2P-Kt: A logic-based ecosystem for symbolic AI. SoftwareX **16**, 100817:1–7 (Dec 2021). https://doi.org/10.1016/j.softx.2021.100817
27. Ciatto, G., Calegari, R., Omicini, A.: Lazy stream manipulation in Prolog via backtracking: The case of 2p-kt:. In: Faber, W., Friedrich, G., Gebser, M., Morak, M. (eds.) Logics in Artificial Intelligence, Lecture Notes in Computer Science, vol. 12678, pp. 407–420. Springer (2021). https://doi.org/10.1007/978-3-030-75775-5_27, 17th European Conference, JELIA 2021, Virtual Event, May 17–20, 2021, Proceedings
28. Ciatto, G., Calegari, R., Siboni, E., Denti, E., Omicini, A.: 2P-Kt: logic programming with objects & functions in Kotlin. In: Calegari, R., Ciatto, G., Denti, E., Omicini, A., Sartor, G. (eds.) WOA 2020 – 21th Workshop "From Objects to Agents". CEUR Workshop Proceedings, vol. 2706, pp. 219–236. Sun SITE Central Europe, RWTH Aachen University, Aachen, Germany (Oct 2020), `http://ceur-ws.org/Vol-2706/paper14.pdf`
29. CompuLaw: Home page. `https://cordis.europa.eu/project/id/833647` (2021), accessed on 23 August 2021
30. Craandijk, D., Bex, F.: Deep learning for abstract argumentation semantics. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Confer-

ence on Artificial Intelligence, IJCAI 2020. pp. 1667–1673. IJCAI.org (2020). https://doi.org/10.24963/ijcai.2020/231
31. Craven, R., Toni, F.: Argument graphs and assumption-based argumentation. Artificial Intelligence **233**, 1–59 (2016). https://doi.org/10.1016/j.artint.2015.12.004
32. Craven, R., Toni, F., Williams, M.: Graph-based dispute derivations in assumption-based argumentation. In: Theory and Applications of Formal Argumentation. Lecture Notes in Computer Science, vol. 8306, pp. 46–62. Springer (2013). https://doi.org/10.1007/978-3-642-54373-9_4
33. Denti, E., Omicini, A., Ricci, A.: Multi-paradigm Java-Prolog integration in tuProlog. Science of Computer Programming **57**(2), 217–250 (2005). https://doi.org/10.1016/j.scico.2005.02.001
34. Dragoni, M., Villata, S., Rizzi, W., Governatori, G.: Combining natural language processing approaches for rule extraction from legal documents. In: AI Approaches to the Complexity of Legal Systems. Lecture Notes in Computer Science, vol. 10791, pp. 287–300. Springer (2017). https://doi.org/10.1007/978-3-030-00178-0
35. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77**(2), 321–358 (1995). https://doi.org/10.1016/0004-3702(94)00041-X
36. Dung, P.M., Thang, P.M.: Fundamental properties of attack relations in structured argumentation with priorities. Artificial Intelligence **255**, 1–42 (2018). https://doi.org/10.1016/j.artint.2017.11.002
37. Dvorák, W., Rapberger, A., Wallner, J.P., Woltran, S.: ASPARTIX-V19 - an answer-set programming based system for abstract argumentation. In: Foundations of Information and Knowledge Systems—11th International Symposium. Lecture Notes in Computer Science, vol. 12012, pp. 79–89. Springer (2020). https://doi.org/10.1007/978-3-030-39951-1
38. d'Avila Garcez, A.S., Besold, T.R., De Raedt, L., Földiák, P., Hitzler, P., Icard, T., Kühnberger, K., Lamb, L.C., Miikkulainen, R., Silver, D.L.: Neural-symbolic learning and reasoning: Contributions and challenges. In: 2015 AAAI Spring Symposia. AAAI Press (2015). https://doi.org/10.13140/2.1.1779.4243
39. García, A.J., Prakken, H., Simari, G.R.: A comparative study of some central notions of ASPIC+ and DeLP. Theory and Practice of Logic Programming **20**(3), 358–390 (2020). https://doi.org/10.1017/S1471068419000437
40. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming **4**(1-2), 95–138 (2004). https://doi.org/10.1017/S1471068403001674
41. van Gijzel, B., Prakken, H.: Relating Carneades with abstract argumentation via the ASPIC$^+$ framework for structured argumentation. Argument & Computation **3**(1), 21–47 (2012). https://doi.org/10.1080/19462166.2012.661766
42. Gordon, T.F., Governatori, G., Rotolo, A.: Rules and norms: Requirements for rule interchange languages in the legal domain. In: Governatori, G., Hall, J., Paschke, A. (eds.) Rule Interchange and Applications, International Symposium, RuleML 2009. Lecture Notes in Computer Science, vol. 5858, pp. 282–296. Springer (2009). https://doi.org/10.1007/978-3-642-04985-9
43. Gordon, T.F., Prakken, H., Walton, D.: The Carneades model of argument and burden of proof. Artificial Intelligence **171**(10-15), 875–896 (2007). https://doi.org/10.1016/j.artint.2007.04.010
44. Gordon, T.F., Walton, D.: Formalizing balancing arguments. In: Computational Models of Argument. Frontiers in Artificial Intelligence and Applications, vol. 287, pp. 327–338. IOS Press (2016). https://doi.org/10.3233/978-1-61499-686-6-327

45. Kok, E.M., Meyer, J.J.C., Prakken, H., Vreeswijk, G.A.: Testing the benefits of structured argumentation in multi-agent deliberation dialogues. In: 11th International Conference on Autonomous Agents and Multiagent Systems. vol. 3, pp. 1411–1412. International Foundation for Autonomous Agents and Multiagent Systems, Valencia, Spain (2012). https://doi.org/10.5555/2343896.2344033
46. Kröll, M., Pichler, R., Woltran, S.: On the complexity of enumerating the extensions of abstract argumentation frameworks. In: Sierra, C. (ed.) 26th International Joint Conference on Artificial Intelligence (IJCAI 2017). pp. 1145–1152. IJCAI.org (2017). https://doi.org/10.24963/ijcai.2017/159
47. Lehtonen, T., Wallner, J.P., Järvisalo, M.: From structured to abstract argumentation: Assumption-based acceptance via AF reasoning. In: Symbolic and Quantitative Approaches to Reasoning with Uncertainty. Lecture Notes in Computer Science, vol. 10369, pp. 57–68. Springer (2017). https://doi.org/10.1007/978-3-319-61581-3
48. Lehtonen, T., Wallner, J.P., Järvisalo, M.: An answer set programming approach to argumentative reasoning in the ASPIC+ framework. In: 17th International Conference on Principles of Knowledge Representation and Reasoning. pp. 636–646 (2020). https://doi.org/10.24963/kr.2020/63
49. Maudet, N., Parsons, S., Rahwan, I.: Argumentation in multi-agent systems: Context and recent developments. In: Maudet, N., Parsons, S., Rahwan, I. (eds.) Argumentation in Multi-Agent Systems, Third International Workshop, ArgMAS 2006, Hakodate, Japan, May 8, 2006, Revised Selected and Invited Papers. LNCS, vol. 4766, pp. 1–16. Springer (2006). https://doi.org/10.1007/978-3-540-75526-5_1
50. Modgil, S., Prakken, H.: The $ASPIC^{+}$ framework for structured argumentation: a tutorial. Argument & Computation **5**(1), 31–62 (2014). https://doi.org/10.1080/19462166.2013.869766
51. Niskanen, A., Järvisalo, M.: μ-toksia: An efficient abstract argumentation reasoner. In: 17th International Conference on Principles of Knowledge Representation and Reasoning. pp. 800–804 (2020). https://doi.org/10.24963/kr.2020/82
52. Omicini, A.: Not just for humans: Explanation for agent-to-agent communication. In: Vizzari, G., Palmonari, M., Orlandini, A. (eds.) AIxIA 2020 DP — AIxIA 2020 Discussion Papers Workshop. AI*IA Series, vol. 2776, pp. 1–11. Sun SITE Central Europe, RWTH Aachen University, Aachen, Germany (Nov 2020), `http://ceur-ws.org/Vol-2776/paper-1.pdf`
53. Omicini, A., Calegari, R.: Injecting (micro)intelligence in the IoT: Logic-based approaches for (M)MAS. In: Lin, D., Ishida, T., Zambonelli, F., Noda, I. (eds.) Massively Multi-Agent Systems II, Lecture Notes in Computer Science, vol. 11422, chap. 2, pp. 21–35. Springer (May 2019). https://doi.org/10.1007/978-3-030-20937-7_2
54. Omicini, A., Mariani, S.: Agents & multiagent systems: En route towards complex intelligent systems. Intelligenza Artificiale **7**(2), 153–164 (Nov 2013). https://doi.org/10.3233/IA-130056
55. Ossowski, S. (ed.): Agreement Technologies, Law, Governance and Technology Series, vol. 8. Springer Netherlands (2012). https://doi.org/10.1007/978-94-007-5583-3
56. Pisano, G., Calegari, R., Omicini, A., Sartor, G.: Arg-tuProlog: A tuProlog-based argumentation framework. In: Calimeri, F., Perri, S., Zumpano, E. (eds.) CILC 2020 – Italian Conference on Computational Logic. Proceedings of the 35th Italian Conference on Computational Logic. CEUR Workshop Proceedings, vol. 2719, pp.

51–66. Sun SITE Central Europe, RWTH Aachen University, CEUR-WS, Aachen, Germany (13-15 Oct 2020), http://ceur-ws.org/Vol-2710/paper4.pdf
57. Podlaszewski, M., Caminada, M., Pigozzi, G.: An implementation of basic argumentation components. In: 10th International Conference on Autonomous Agents and Multiagent Systems. vol. 3, pp. 1307–1308. International Foundation for Autonomous Agents and Multiagent Systems (2011). https://doi.org/10.5555/2034396.2034539
58. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation **1**(2), 93–124 (2010). https://doi.org/10.1080/19462160903564592
59. Riveret, R., Rotolo, A., Sartor, G.: A deontic argumentation framework towards doctrine reification. Journal of Applied Logics—IfCoLog Journal of Logics and their Applications **6**(5), 903–940 (2019), https://collegepublications.co.uk/ifcolog/?00034, Special Issue: Reasoning for Legal AI
60. Snaith, M., Reed, C.: TOAST: online ASPIC$^+$ implementation. In: Computational Models of Argument. Frontiers in Artificial Intelligence and Applications, vol. 245, pp. 509–510. IOS Press (2012). https://doi.org/10.3233/978-1-61499-111-3-509
61. Toni, F.: A generalised framework for dispute derivations in assumption-based argumentation. Artificial Intelligence **195**, 1–43 (2013). https://doi.org/10.1016/j.artint.2012.09.010
62. Toni, F.: A tutorial on assumption-based argumentation. Argument & Computation **5**(1), 89–117 (2014). https://doi.org/10.1080/19462166.2013.869878
63. Vreeswijk, G.: Abstract argumentation systems. Artificial Intelligence **90**(1–2), 225–279 (1997). https://doi.org/10.1016/S0004-3702(96)00041-0