Arg-tuProlog: A tuProlog-based Argumentation Framework ⋆ ⋆⋆

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

[1] ALMA–AI Interdepartmental Center of Human Centered AI, Alma Mater Studiorum–Università di Bologna, Italy
[2] Dipartimento di Informatica – Scienza e Ingegneria (DISI), Alma Mater Studiorum–Università di Bologna, Italy

Abstract. Over the last decades, argumentation has become increasingly central as a frontier research within artificial intelligence (AI), especially around the notions of *interpretability* and *explainability*, which are more and more required within AI applications. In this paper we present the first prototype of Arg-tuProlog, a logic-based argumentation tool built on top of the tuProlog system. In particular, Arg-tuProlog enables *defeasible reasoning* and *argumentation*, and deals with *priorities* over rules. It also includes a formal method for dealing with burden of proof (burden of persuasion). Being lightweight and compliant to the requirements for *micro-intelligence*, Arg-tuProlog is perfectly suited for injecting argumentation into distributed pervasive systems.

Keywords: Argumentation · logic-based argumentation · burden of persuasion · tuProlog · micro-intelligence · symbolic intelligence.

1 Introduction

In recent years we are witnessing a renewed interest in logical models for symbolic automated reasoning—mostly related to their interpretability and explainability features, more and more required by *artificial intelligence* (AI) technologies [7]. In a landscape where an ever-growing number of computational agents situated in real-life contexts take autonomous decisions, acting on behalf of people, *argumentation* theories and technologies play a key role: they may contribute to rational decision-making, facilitate agreements, contribute to the resolutions of

⋆ G. Pisano has been supported by the European Union's Justice Project "InterLex: Advisory and Training System for Internet-related private International Law" (G.A. 800839). R. Calegari and G. Sartor have been supported by the H2020 ERC Project "CompuLaw" (G.A. 833647). A. Omicini has been supported by the H2020 Project "AI4EU" (G.A. 825619).

disputes and provide explanations [6]. Logical argumentation is indeed a well-known and established paradigm, mainly used for achieving effective communication and coordination between agents [16].

As intelligent systems need to communicate, reason, and act in accordance with the law, computable models of legal argumentation are particularly significant for the development of robust and law compliant intelligent systems [16, 2]. Unfortunately, a mature environment for argumentation-based approaches – fruitfully integrated with MAS and legal models – still does not exist [8]. This preliminary work aims to contribute to filling this void by presenting a light-weight argumentation tool especially designed for the above-mentioned context. In particular, the tool is designed according to the definition of *micro-intelligence* in [19, 5], whose key features are *(i)* customisation of the inference methods – as deduction, abduction, argumentation, just to name a few – to be exploited opportunistically in an integrated and easily-interchangeable way, *ii)* situated-ness – i.e., the awareness and reactiveness to the surrounding environment, such as the normative institution – and, *iii)* ability to act at the micro-level of the system, so as to be easily injectable in disparate contexts and architectures.

Accordingly, this work presents the Arg-tuProlog tool—a lightweight tuProlog-based implementation for structured argumentation [3] in compliance with the micro-intelligence definition. The model adopts an ASPIC$^+$-like syntax [20] and has been extended to capture the burden of persuasion issues according to [9]. Also, deontic extensions [21] – i.e., explicit representation of obligations and permissions – are included.

The work is structured as follows. In Section 2, we provide a global overview of the framework discussing the formal foundations and the argumentation language for the operational use of the tool. Then, Section 3 discusses the architecture and the engine interface, whereas Section 4 presents the directives introduced by the Arg-tuProlog framework. Section 5 discusses a working example of the system. Finally, in Section 6 we overview similar systems already proposed in the literature. Section 7 provides for the final remarks.

2 Framework Model & Language

2.1 Formal model

Arg-tuProlog is a modular rule-based argumentation system to represent, reason, and argument upon conditional norms featuring obligations, prohibitions, and (strong or weak) permissions also according to burden of persuasion constraints.

The approach is based on common constructs in computational models of argument that lay their root in Dung's abstract argumentation [14] and structured argumentation [3]: rule-based arguments, argumentation graphs, argument labelling semantics and statement labelling semantics. Arguments are formed by chaining applications of inference rules into inference trees or graphs—i.e., arguments are constructed using deductive inference rules that licence deductive inferences from premises to conclusions. As in ASPIC$^+$, arguments are defined

relatively to an argumentation theory AT by chaining applications of the inference rules starting with elements from the knowledge base. All the formulas in the knowledge base are called *premises* and are used to build the arguments; the reached supported argument is denoted as its *conclusion*; the last inference rule used in the argument is called *top rule*.

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 using some semantics. For our purposes, we resort to labelling semantics as reviewed in [1]. Accordingly, we endorse {IN, OUT, UND}-labellings where each argument is associated with one label which is either IN, OUT, or UND, respectively meaning that the argument is accepted, rejected, or undecided. Arguments and attack relations can be then captured in Dung's abstract argumentation graphs, originally called abstract argumentation frameworks in [14]. Formal accounts of the adopted deontic extensions are discussed in detail in [21], while the implemented burden of persuasion model can be found in [9]. With respect to the latter, Arg-tuProlog presents a first technology reification for the burden models.

2.2 Language

The language to encode knowledge in the argumentation framework is of primary concern. In fact, we aim at providing users with an interface as plain as possible in terms of interpretability and human readability.

Arg-tuProlog adopts an ASPIC$^+$-like syntax – introduced in [20, 17] – made of two main elements: defeasible rules and preferences over rules. With respect to the original notation, the Arg-tuProlog language has been extended with a specific notation for dealing with burden of persuasion and deontic operators.

Defeasible rules, facts, and deontic expressions. This element of the language allows facts and rules to be encoded. Statements take the form

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

Unconditioned statement (facts) can be expressed with the notation:

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

Premises and conclusions can take any form accepted in Prolog: i.e., all Prolog terms – e.g. atoms, variables, lists, compound terms – are allowed, with the addition of three further notations

- $p(term)$ to indicate permission;
- $o(term)$ to indicate obligation.
- $-term$, to indicate a strong negation, as opposed to the negation as failure implemented within the tuProlog engine.

The formalism includes permissions, prohibitions, and obligations; the corresponding logic is captured by defeasible rule schemata, modelling basic deontic inference [21].

In general, strong negations cannot be nested. The only exception to this rule is in the use of deontic operators, in particular to manage prohibitions—a lack of prohibition can be defined exploiting two strong negations: $-o(-term)$.

Finally, inside premises, weak negation (negation as failure) can be expressed as $\sim (term)$. As above, this operator can not be nested. This notation is useful to express exceptions to the applicability of a rule or make assumption: it defines an undercut attack.

Superiority relation. Many non-monotonic reasoning and argumentation tools – ABA+ [12], Defeasible Logic [18], ASPIC$^+$ [17] – offer the possibility to specify superiority between rules. Preferences over rules – possibly regarding the reliability of the statements – are a fundamental element in AI scenarios in which conflicting and inaccurate knowledge is the normal state of affair.

Arg-tuProlog makes it possible to denote preferences by using the following notation: $sup(ruleName_1, ruleName_2)$ This proposition states that the rule with identifier equal to $ruleName_1$ is superior to the one with identifier $ruleName_2$. The superiority relation between $ruleName_1$ and $ruleName_2$ makes it possible to solve the conflict between arguments in favour of arguments built from $ruleName_1$. The preference relation over arguments can be defined in various ways based on the preference over rules. We adopt a simple last-link ordering, according to which an argument A is preferred over another argument B if and only if the top rule of A is superior to the rule top rule of B.

Burden of persuasion. The burden of persuasion, born in the legal field but then extended also to philosophy, allows a response on the acceptability of an argument to be obtained even in situations 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 evidences are provided. 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. Generally speaking, we can say that if there is a burden of persuasion for a conclusion ϕ, then argument for ϕ will be rejected unless they strictly prevail against all counterarguments (that are not rejected on other grounds). As a consequence, the collision between arguments for ϕ and arguments for $\bar{\phi}$ will be decided in favour of the latter, in the absence of a priority for the first.

In our model, the burden of persuasion may be captured in a second stage concerning argument labellings, only after the base labelling has been computed. It only concerns the arguments labelled UND. So, a burden of persuasion labelling addresses UND arguments, and assigns OUT and IN labels depending on the allocation of the burden of persuasion.

Formally, 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.

3 Engine Design & Architecture

The Arg-tuProlog framework[3] aims at providing a comprehensive and innovative tool for spreading intelligence and argumentation capabilities in nowadays challenging AI context. It tries to address the issues arising in the actual technological landscape of intelligent systems—with particular attention to the legal and explainability issues.

The interoperability and portability requirements of intelligent systems drive us to the choice of tuProlog [13] as the main technological foundation. The Kotlin-based engine – devoted to heavy-interconnected and pervasive contexts – enables the system to run in the more disparate environments.

The main and distinguishing aspect of the Arg-tuProlog engine is represented by its design, which aims at providing two distinct ways of use:

a) *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;
b) *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 Arg-tuProlog framework a fit choice for the aforementioned AI pervasive scenarios.

In the context of multi-agent systems (MAS), Arg-tuProlog enables agents to correctly deal with incomplete or missing knowledge, encouraging argumentation and dialogue among them in order to reach an agreement or influence a decision. In our vision, the agent intelligent behaviour is likely to be associated with the capability of debating about the current context, by reaching a consensus on what is happening around and what is needed, and by triggering and directing proper conversations to decide how to collectively act in order to reach the future desirable state of the affairs. Knowledge is shared and decentralised, and the Arg-tuProlog engine provides both the agents and the environment abstraction of MAS with reasoning and argumentation capabilities, making it possible to reach a social consensus on what is better to do. For these reasons, it fits perfectly with the AI scenarios recalled in Section 1.

[3] https://pika-lab.gitlab.io/argumentation/arg2p/

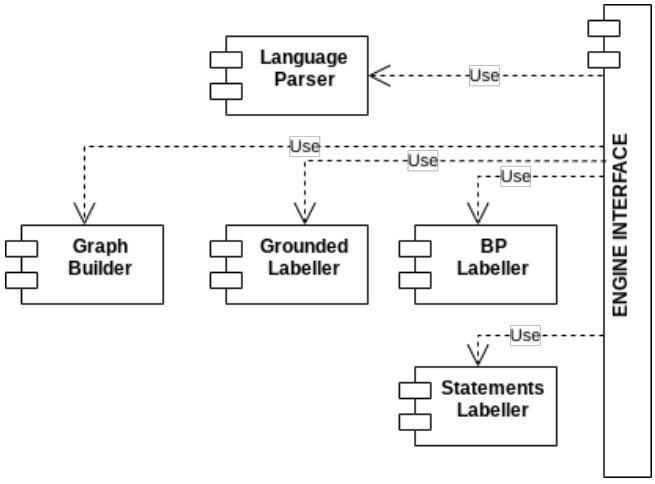


Fig. 1: The system modules diagram.

3.1 Architecture & API

The framework leverages on the underlying tuProlog engine. All the required components exploit the tuProlog feature to allow external libraries to be included during the evaluation process. Consequently, the entire framework is a collection of tuProlog compatible libraries.

Fig. 1 shows the modules composing the system. It is designed in a fully-modular way. Every function inside the framework – e.g., graph building, or argument labelling – is sealed in the corresponding module. Modularity highly improves upgradability and flexibility – in terms of adding of new features or requirements modification – of the entire system. For example, one may consider the *Grounded Labeller*, responsible for computing the grounded labelling of the argumentation graph. In order to add a different labelling semantics, it would only require to create an ad-hoc module, without impacting the rest of the system. In the following, we describe each component in detail.

Engine interface. The *Engine Interface* module hides all the complexity of the framework. It exposes only the two usage predicates—namely, *answerQuery/2* and *buildLabelSets/0* transparently orchestrating the basic modules.

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—namely, IN, OUT, and UND. The IN set includes facts classified as acceptable, the OUT is for the rejected ones and the UND is for those of which it was not possible to classify due to lack of information. For instance, assume to query the system about doctors' liability (for instance, in case of medical malpractice, according to the applicable law to which doctors are liable for the harm suffered by a patient if they were negligent in treating

the patient) in a case where Dr. Murphy should be considered liable, while Dr. House not. The predicate usage would be `answerQuery(liable(Doctor), Yes, No, Und)`, and the result would be composed by the three sets containing the solutions IN=`[liable(drMurphy)]`, OUT=`[liable(drHouse)]` and UND=`[]`.

The predicate `buildLabelSets` builds and prints (in the output interface) the argument and the statement labellings according to the provided theory based on the construction and evaluation of the arguments. All arguments and statements are therefore evaluated.

The user interface also provides two flags, which can be set in the application as facts of the Prolog theory, for setting useful options in the resolution process. Both flags deals with the management of the burden of persuasion. The first flag is `demonstration`: if activated, the output prints the labelling resolution process step by step, pointing out the specific part of the definition presented in [9] led to the resulting label for each argument. The second flag that the user can set is `disableBPcompletition`, which makes it possible to get a label conforming to the definition with no completion of the labelling—i.e., which may not be complete.

The intuition behind incompleteness is the following: when we accept a conclusion ϕ since the burden of persuasion is on $\bar{\phi}$, we may still remain uncertain on the premises for that conclusion. For instance, we may accept that the doctor was negligent since he failed to provide a convincing argument about why he was not negligent, and still be uncertain whether the doctor did or did not comply with the guidelines. The opposite approach is also possible: if we accept an argument based on the burden of persuasion, we are also bound to accept all of its sub-arguments. This approach leads to the concepts of a complete and a grounded burden of persuasion labelling [9].

Language parser. The *Language Parser* module is aimed at converting the rules in the defined argumentation language (Section 2) into a correct Prolog theory. In fact, in this first prototype, Arg-tuProlog acts as a meta-interpreter. The meta-interpreter accepts theories in a well-defined argumentation language, translates them in Prolog – exploiting this module – and, finally, completes the computation providing the requested solutions.

Algorithmic modules. The rest of the modules are mainly linked to algorithmic responsibilities, in particular:

- *Graph Builder* builds the argumentation graph
- *Grounded Labeller*, in charge of computing grounded labelling of the argumentation graph, according to the Dung's notions of grounded 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 the two previous steps.

Details about these modules are provided in Section 4.

4 Operational Interface

In Section 3 we present the two user predicates provided by the Arg-tuProlog framework—namely, *answerQuery*/4 and *buildLabelSets*/0. Their semantics leverages on a common resolution process exploited to build the argumentation graph and evaluate the arguments. In particular, the resolution process can be split into three distinct steps:

1. the argumentation graph construction;
2. the arguments labelling according to the different labelling semantics—namely, grounded, burden of persuasion (BP), and BP complete;
3. the statements labelling according to the arguments labelling.

The predicate *buildLabelSets*/0 performs all the three steps and provide the argumentation graph and its labelling as solutions. The predicate *answerQuery*/4, once built the argumentation graph and its labelling, searches for the specific solution (input `Goal`) in the labelling sets. In the following, further details of each step are discussed.

4.1 Argumentation graph construction

As in any logic-based argumentation framework, an argument is not a generic entity: instead, it is a pair, whose first part is a coherent minimum set of formulas sufficient to prove the formula contained in its second part. Relationships of support – in the case an argument sustains the same conclusion of another one – or attack – in the case an argument conclusion is in contrast with another argument claim – connect the arguments between them. Construction of arguments, and of their relationships, is ascribed to the predicate *buildArgumentationGraph*/1. Formally, arguments have three properties (c.f. [17]):

a) a unique identifier, given by the set of identifiers (*ruleName*, *RN* in the following) of the rules applied for its creation;
b) the identifier of the last applied rule—i.e., its top rule (*TopRN*);
c) the conclusion supported by this argument (*Conc*).

They are added to the theory in the form:

$$argument([[RN_1, \ldots, RN_n], Top\ RN, Conc]).$$

Arguments are built searching all the distinct and coherent subsets of facts and rules sharing the same conclusion contained by the input theory. The support relationship between arguments is expressed as follows:

$$support(supporter\_argument, supported\_argument).$$

Through the predicate *support*/2, it is possible to reconstruct the entire tree that led to the creation of an argument.

The attack relations between the arguments – rebut or undercut – are instead defined in the form:

$$attack(attacker\_argument, attacked\_argument).$$

4.2 Argument labelling

Once the argumentation framework is built, it can be evaluated, and labels can be associated with each argument according to a specified semantics. Labels are a way of expressing acceptance/rejection of the arguments.

In Arg-tuProlog, labelling is built in two stages. The first stage is aimed at computing the equivalent of Dung's grounded extension [14]. Intuitively, this unique extension requires an argument to be accepted in all the existent complete extensions and the number of IN labels to be minimised. The operator used for this purpose is *argumentLabelling*/2. This operator requires as input the set of arguments, supports and attacks – built in the previous step – and returns the three label sets IN, OUT, and UND as output.

The second stage of labelling takes into account the possible burden of persuasion (BP) on propositions. First, the BP specifications are pre-processed by grounding the contained variables if present (taking into account only evaluable arguments). The new set of grounded BP terms is then added to the theory in the form:

$$reifiedBp([[grounded\_term_1], \ldots, [grounded\_term_n]]).$$

Then, the *argumentBPLabelling*/2 predicate is called to perform the labelling according to the semantics specified in [9]. Both BP-labelling and completion of BP-labelling – in according to the formal model – can be performed by the *argumentBPLabelling*/2 predicate. The option to differentiate the operation modality is provided by a flag that can be set by the user as a fact in the theory—namely, *disableBPcompletition*.

4.3 Statement labelling

The predicate *statementLabelling*/2 labels all the statements in a set of conclusions of an argument according to the label of the argument. Therefore, the conclusions of IN arguments are labelled as IN, and the same for OUT and UND.

5 Example

To ground the discussion, and show the Arg-tuProlog framework effectiveness, in the following we discuss a running example from [9].

Let us consider a case in civil law where the allocation of the burden of persuasion act as follows. We assume that a doctor is liable for the harm suffered by a patient. However, there is an exception: the doctor can avoid liability if he shows he was not negligent. We also assume that a doctor is considered to be not negligent if he followed the medical guidelines that govern the case. Finally, we assume that it is uncertain whether the doctor has followed the guidelines in the case at hand. Let us first model the case without taking the burden into account. Accordingly, let us suppose the following premises and rules:

$f1$: $\Rightarrow \neg guidelines$ $f2$: $\Rightarrow guidelines$
$f3$: $\Rightarrow harm$
$r1$: $\neg guidelines \Rightarrow negligent$ $r2$: $guidelines \Rightarrow \neg negligent$
$r3$: $\neg negligent \Rightarrow \neg liable$ $r4$: $harm \Rightarrow liable$

with $r3 \succ r4$. The theory can be expressed in the Arg-tuProlog argumentation language as follows:

```
f1 : [] => -guidelines.
f2 : [] => guidelines.
f3 : [] => harm.
r1 : -guidelines => negligent.
r2 : guidelines => -negligent.
r3 : -negligent => -liable.
r4 : harm => liable.
sup(r4, r3).
```

According to the knowledge base, we can then build the following arguments:

$\mathsf{A}_{\mathsf{f1}}: \Rightarrow \neg guidelines$ $\mathsf{B}_{\mathsf{f2}}: \Rightarrow guidelines$ $\mathsf{C}_{\mathsf{f3}}: \Rightarrow harm$
$\mathsf{A}_{\mathsf{r1,f1}}: \mathsf{A}_{\mathsf{f1}} \Rightarrow negligent$ $\mathsf{B}_{\mathsf{r2,f2}}: \mathsf{B}_{\mathsf{f2}} \Rightarrow \neg negligent$ $\mathsf{C}_{\mathsf{r4,f3}}: \mathsf{C}_{\mathsf{f3}} \Rightarrow liable$
$\mathsf{B}_{\mathsf{r3,r2,f1}}: \mathsf{B}_{\mathsf{r2,f2}} \Rightarrow \neg liable$

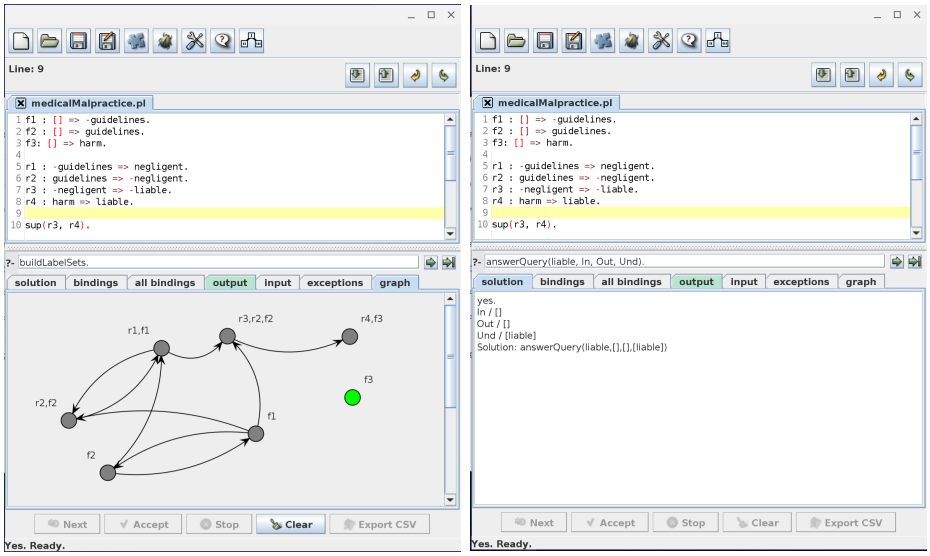
The argumentation graph and its grounded $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$-labelling are depicted in Fig. 2a, in which all arguments are UND, except argument C_{f3}. The green nodes are the ones labelled as IN, the red ones as OUT, the grey as UND. Fig. 2b depicts the same situation but a difference operation mode of the Arg-tuProlog engine. In particular, the system is asked to prove the `liable` goal, and as show in the solution tab, `liable` has been labelled as UND, since in the current situation no reasoning about the doctor liability can be done.

The result is not satisfactory according to the law, since it does not take into account the applicable burdens of persuasion. The doctor should have lost the case – i.e., be declared liable – since he failed to discharge his burden of proving he was not negligent—namely, he didn't satisfy his burden of persuasion for his non-negligence. The doctor's failure results from the fact that it remains uncertain whether he followed the guidelines. In order to capture this aspect of the argument, we need to provide an indication for the burden of persuasion.

Now we assume – according to the Italian Law – to have $bp(harm)$ and $bp(\neg negligent)$, i.e., the doctor has to persuade the judge that he was not negligent, and the patient has to persuade the judge that he was harmed. We also assume it remains uncertain whether the doctor was diligent since it is uncertain whether he followed the guidelines, while the patient succeeded in proving harm (as the claim that there is harm was unchallenged). Then, since the doctor failed to meet his burden of proof, the doctor is considered to be liable.

The grounded BP-labelling is depicted in Fig. 3. In particular, Fig. 3a shows the argumentation graph, Fig. 3b the corresponding arguments labelling, and Fig. 3c the labelling resolution process step by step.

In contrast with the grounded $\{\mathsf{IN}, \mathsf{OUT}, \mathsf{UND}\}$-labelling where every argument is labelled UND, arguments $\mathsf{A}_{\mathsf{r1,f1}}$ and $\mathsf{C}_{\mathsf{r4,f3}}$ are now labelled IN, while $\mathsf{B}_{\mathsf{r2,f2}}$ and



(a) Grounded argumentation graph (b) Query-based mode

Fig. 2: The Arg-tuProlog IDE with no burden of persuasion: grounded argumentation graph (a) and example of query-based mode usage (b).

$\mathsf{B}_{\mathsf{r3,r2,f1}}$ are labelled OUT. It is still uncertain whether the doctor followed the guidelines (the corresponding arguments A_{f1} and B_{f2} are UND).

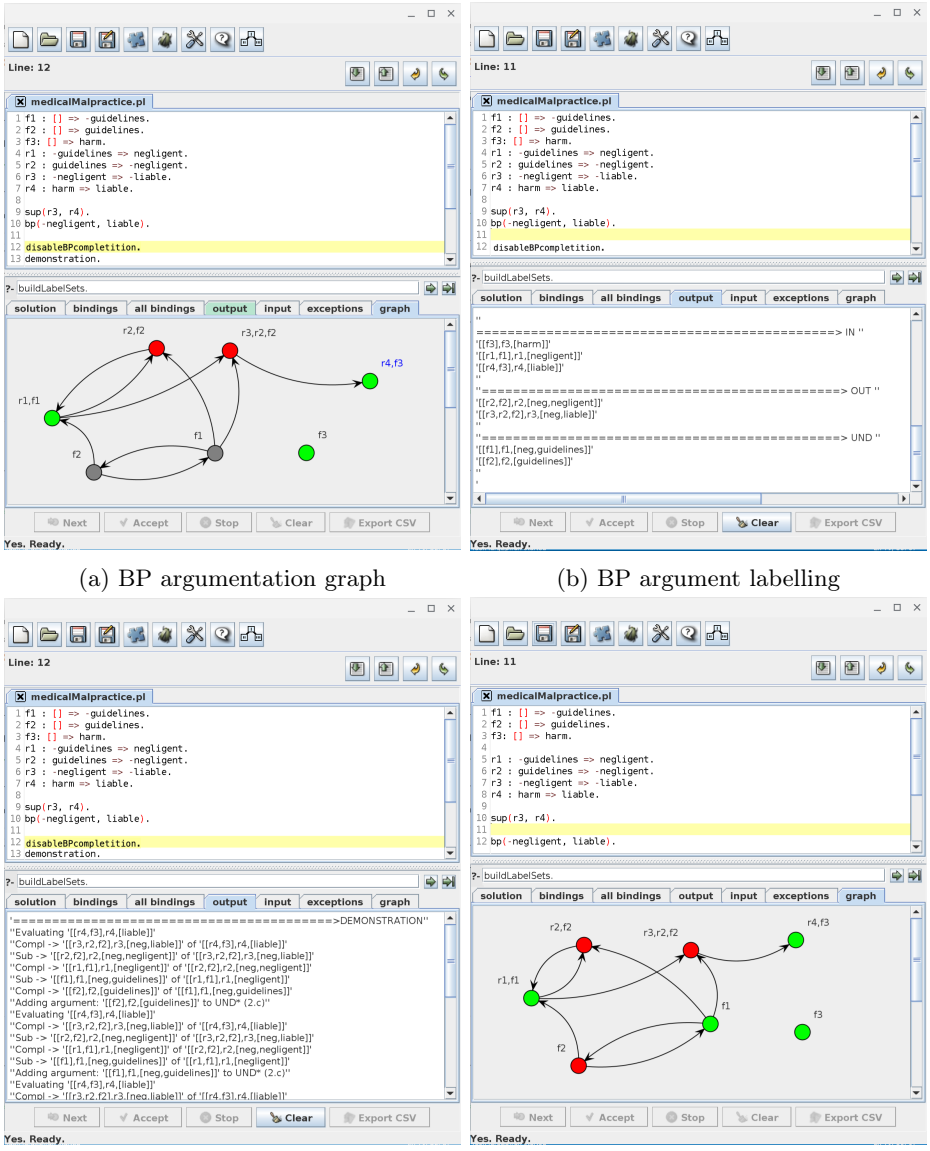
A different result can be obtained by enabling the completion of the BP-labelling. Consequently, as depicted in Fig. 3d, argument A_{f1} for $\neg$*guidelines* is IN and argument B_{f2} for *guidelines* is OUT.

6 Related works

Many argumentation tools have been proposed along the last decades and, in this work, we do not mean to give an exhaustive survey. In order to strengthen the motivation for this work, in the following we recall the related research panorama, highlighting differences and similarities. We restrict our scope to argumentation *technologies* only, that is, those coming with downloadable and usable tools—as in the case of Arg-tuProlog.

Three main approaches can be distinguished in the available tools: the area of the rule-based abstract argumentation – where ASPIC$^+$ [17] is the most known –, the area of assumption-based argumentation – where ABA$^+$ [10] can be classified as the reference formalism – and DeLP [15], based on the use of defeasible logic. Although there is a strong convergence of different systems for defeasible reasoning, some distinctions may be relevant to different application domains.

The possibility of using open (non-ground) rules in knowledge, and of using different instances of the same predicates in different rules, could be a key ad-



(a) BP argumentation graph

(b) BP argument labelling

(c) BP labelling resolution process

(d) BP-labelling graph completation

Fig. 3: The Arg-tuProlog IDE with burden of persuasion: the picture depicts the IDE usage in case of burden of persuasion highlighting the use of the user flags.

vantage, especially when the same rule has to be applied to different instances within a single argument. This is usually not allowed in ABA-based tools, and we consider this a key feature in the design of Arg-tuProlog.

When a system has to deal with a high number of uncertain conflicts, the ability to rely not only on sceptical, but also on credulous reasoning, and in general on different semantics, may be important. Argumentation approaches have this ability natively while the one based on defeasible logic usually not, though also ambiguity propagation in defeasible logic can lead to similar results. The Arg-tuProlog prototype currently implements only the grounded semantics.

When a system has to address complex issues of legal reasoning, and full explainability is required, the ability to provide a picture of existing arguments and of the relations between them – with an an explanation on which arguments should or could be finally endorsed – may become a critical feature. This is a feature we could find in structured argumentation tools, like Arg-tuProlog, but not in defeasible logic tools.

A very similar approach to our work that deserves to be mentioned is Argue tuProlog [4]. Very similar in terms of purposes and architecture, the Argue tuProlog prototype is not practically usable, since it is no longer maintained and based on old versions of tuProlog. Furthermore, Argue tuProlog vision is not tailored to pervasive intelligent systems: its main goal, unlike Arg-tuProlog is not to act as a distributor of symbolic intelligence, suitably integrated with sub-symbolic techniques. On the other hand, in the same way as Arg-tuProlog, Argue tuProlog can perform query-based reasoning. In Arg-tuProlog solving the acceptance of a given statement implies the construction of the whole argumentation graph—one of the main shortcomings of the prototype. Conversely, in Argue tuProlog the acceptance of a single statement is computed using an ad hoc dialogical procedure similar to the *dispute derivation* algorithm available in ABA systems [22, 11]. In a nutshell, two actors – namely the proponent and the opponent – discuss sharing arguments and counterarguments relatively to the claim to be evaluated and try to contradict each other—for instance, finding a valid counterargument or undercutting some premises.

From a technological perspective, many improvements are required in order to make existing tools usable and effective in a distributed environment, as well as easily downloadable/deployable and well documented. Indeed, almost all the available systems can be classified as early prototypes, lacking in most cases of any support or documentation.

Finally, all the above reflections have to be combined with considerations about the effectiveness and ease in the use of the argumentation tools provided.

7 Discussion & Conclusions

The work shows the effectiveness of Arg-tuProlog to: *(i)* deal with inconsistent information—thus enabling defeasible reasoning; *(ii)* integrate legal aspects—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 greatest strengths of the tool lies in its architecture: both modularity and complete integration with logic programming systems – being Prolog based – make it highly suitable for the target pervasive contexts of intelligent systems. In such contexts, in fact, it is preferable to be able to hook/unhook functions – i.e., libraries – according to the requirements. Moreover, the intrinsic integration with logic programming also makes it easier to respond to requirements such as interpretability, understandability and explainability. However, the prototype presented in this work is the starting point of the desiderata, various enhancements should be considered both in terms of research and efficiency.

Efficiency issues. The main limit of the prototype is related to the query-based mode. In fact, to answer a single query – on a single statement – the argumentation graph on the whole theory has to be performed. As discussed in Section 6, different techniques have already been developed allowing for efficient single argument evaluation. We mean to exploit these algorithms in theArg-tuProlog framework. The second issue concerns the mechanism for building the argumentation graph. To derive the argumentation trees from facts and rules, an exhaustive search on the knowledge base is performed—i.e., the computational cost is roughly quadratic for the number of rules of the theory. Also, the labelling algorithm based on the burden of persuasion is extremely expensive—quadratic with respect to the number of arguments. Consequently, the most expensive operators should be implemented more efficiently—for example 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 and issues. From the point of view of the research, Arg-tuProlog represents, in our vision, the fundamental brick for distributing symbolic intelligence in intelligent systems in compliance with the concept of micro-intelligence. The enabling of argumentation capabilities should give system actors properties like understandability and explainability – since the actors can argument 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. The high interoperability of Arg-tuProlog and its modular architecture make us think that it could be a useful technology for these purposes, yet many open points still deserve attention.

References

1. 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
2. Bench-Capon, T.J.M., Prakken, H., Sartor, G.: Argumentation in legal reasoning. In: Simari, G.R., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 363–382. Springer (2009). https://doi.org/10.1007/978-0-387-98197-0_18
3. 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
4. Bryant, D., Krause, P.J., Vreeswijk, G.: Argue tuProlog: A lightweight argumentation engine for agent applications. In: Dunne, P.E., Bench-Capon, T.J.M. (eds.) Computational Models of Argument: Proceedings of COMMA 2006, September 11–12, 2006, Liverpool, UK. Frontiers in Artificial Intelligence and Applications, vol. 144, pp. 27–32. IOS Press (2006), http://ebooks.iospress.nl/publication/2929
5. 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), http://amsdottorato.unibo.it/8521/
6. Calegari, R., Ciatto, G., Denti, E., Omicini, A.: Logic-based technologies for intelligent systems: State of the art and perspectives. Information **11**(3), 1–29 (Mar 2020). https://doi.org/10.3390/info11030167
7. Calegari, R., Ciatto, G., Omicini, A.: On the integration of symbolic and sub-symbolic techniques for XAI: A survey. Intelligenza Artificiale **14**(1), 7–32 (2020). https://doi.org/10.3233/IA-190036
8. 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
9. 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.) Proceedings 36th International Conference on Logic Programming (Technical Communications). Electronic Proceedings in Theoretical Computer Science, vol. 325, pp. 151–163. Open Publishing Association (18–24 Sep 2020). https://doi.org/10.4204/EPTCS.325.21
10. 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
11. Craven, R., Toni, F., Williams, M.: Graph-based dispute derivations in assumption-based argumentation. In: Black, E., Modgil, S., Oren, N. (eds.) Theory and Applications of Formal Argumentation - Second International Workshop, TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected papers. LNCS, vol. 8306, pp. 46–62. Springer (2013). https://doi.org/10.1007/978-3-642-54373-9_4
12. Cyras, K., Toni, F.: ABA+: assumption-based argumentation with preferences. In: Baral, C., Delgrande, J.P., Wolter, F. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016. pp. 553–556. AAAI Press (2016), http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12877

13. 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
14. 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
15. 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
16. 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
17. 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
18. Nute, D.: Defeasible logic. In: Proceedings of the 14th International Conference on Applications of Prolog, INAP 2001, University of Tokyo, Tokyo, Japan, October 20-22, 2001. pp. 87–114. The Prolog Association of Japan (2001)
19. 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, LNCS, vol. 11422, chap. 2, pp. 21–35. Springer (May 2019). https://doi.org/10.1007/978-3-030-20937-7_2
20. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-Classical Logics **7**(1), 25–75 (1997). https://doi.org/10.1080/11663081.1997.10510900
21. 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
22. 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