








Article

Argumentation and Defeasible Reasoning in the Law

Marco Billi ^{1,*}, Roberta Calegari ¹, Giuseppe Contissa ¹, Francesca Lagioia ¹, Giuseppe Pisano ¹,
Galileo Sartor ² and Giovanni Sartor ¹

¹ Alma AI—Alma Mater Research Institute for Human-Centered Artificial Intelligence, ALMA MATER STUDIORUM—Università di Bologna, 40100 Bologna, Italy; roberta.calegari@unibo.it (R.C.); giuseppe.contissa@unibo.it (G.C.); francesca.lagioia@unibo.it (F.L.); g.pisano@unibo.it (G.P.); giovanni.sartor@unibo.it (G.S.)

² Computer Science Department, University of Torino, 10124 Torino, Italy; galileo.sartor@unito.it

* Correspondence: marco.billi3@unibo.it

Abstract: Different formalisms for defeasible reasoning have been used to represent knowledge and reason in the legal field. In this work, we provide an overview of the following logic-based approaches to defeasible reasoning: defeasible logic, Answer Set Programming, ABA+, ASPIC+, and DeLP. We compare features of these approaches under three perspectives: the logical model (knowledge representation), the method (computational mechanisms), and the technology (available software resources). On top of that, two real examples in the legal domain are designed and implemented in ASPIC+ to showcase the benefit of an argumentation approach in real-world domains. The CrossJustice and Interlex projects are taken as a testbed, and experiments are conducted with the Arg2P technology.

Keywords: argumentation; defeasible reasoning; tools and technologies; Arg2P



Citation: Billi, M.; Calegari, R.; Contissa, G.; Lagioia, F.; Pisano, G.; Sartor, G.; Sartor, G. Argumentation and Defeasible Reasoning in the Law. *J* **2021**, *4*, 897–914. <https://doi.org/10.3390/j4040061>

Academic Editor: Larisa Ivascu

Received: 31 October 2021

Accepted: 9 December 2021

Published: 18 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the last decades, much work has been done in AI and law research on computable models of the law and the development of legal reasoning systems.

Starting from the assumptions that legal provisions can usually be represented as rules and that legal reasoning is based on inferences based on the application of rules to facts [1], many attempts have been made to develop rule-based systems for the law, including seminal contributions such as [2]. In the course of the last 30 years, theoretical issues concerning the representation of legal knowledge have been discussed, including aspects such as isomorphism [3], deontic positions [4], temporal aspects [4], cased-based reasoning [5], and how to deal with defeasibility in law [6], in particular by means of argumentation-based approaches [7].

Despite the apparent limitations of symbolic approaches and the recent advent of data-driven approaches, logic keeps playing an important role as a knowledge representation scheme to capture both norms and facts. Logic is also useful to complement sub-symbolic and data-driven models with reasoning capabilities [8–10].

Subsequent research adopted a broader perspective, based on argumentation, considering that defeasibility in legal reasoning reflects the dialectics of arguments in judicial proceedings. Each party provides arguments supporting his or her position, and such arguments may conflict with those made by the other party. The debate between parties is usually transferred to the judicial opinion that takes into account the results of the dispute and determines its output. To convincingly justify a judicial decision in a case, it is not sufficient to state a single argument; it is instead necessary to establish that the winning argument prevails over all the contrary arguments or that the latter has to be rejected on other grounds. Indeed, it has been stated that the problem of defeasibility in the law results from having arguments both for and against the application of a rule, which need to be adjudicated between [7].

The legislator itself often suggests how to construct defeaters to certain arguments. For example, to indicate that liability in tort can be excluded by appealing to self-defence or a state of necessity, the legislator may use any of the following formulations:

- Unless clause. One is liable if one voluntarily causes damage, unless one acts in self-defence or in a state of necessity.
- Explicit exception. One is liable if one voluntarily causes damage. One is not liable for damages if one acts in self-defence or in a state of necessity.
- Presumption. One is liable if one voluntarily causes damage and one does not act out of self-defence or a state of necessity. The absence of both is presumed.

All these aspects of the law, i.e., the tension between generic legal concepts and their exceptions, between the general terms of the law and the particulars of a case and its context, and the adversarial nature of legal procedures, make it so that legal reasoning involves the construction of arguments to attack, defend, override, or undercut other arguments [7].

Many frameworks have been developed to capture the dialectical relation between arguments in legal contexts. Such frameworks typically enable the interaction between arguments to be analysed, determining which arguments are justified, or rather overruled, relative to a set of arguments that may attack one another. Among the frameworks developed in AI and law, we can mention those developed by Prakken and Sartor [11], Gordon [12], Hage [13], and Bench-Capon [14].

In recent years, the argument frameworks developed within AI and law have merged with the growing research on computational argumentation [15]. In particular, the semantic originally developed by Dung has been connected to the research by Prakken and Sartor and Bench-Capon. This has led in particular to the development of frameworks for structured argumentation such as ASPIC+ [16] and Carneades [17,18], which are partially inspired by the features of legal reasoning. Recently, the interest in argumentation has also been stimulated by the many attempts to use argumentation to generate an explanation for the outcome of machine learning systems [19,20].

The vast extent of research on computable legal argumentation so far has not been matched by corresponding number of deployed applications. We believe that this depends on the fact that no open development environment suitable for legal applications is so far available for argumentation systems. In the following section, we review some implemented argumentation technologies, present their basic features, and discuss some related issues. Finally, we exemplify the use of one of these technologies—Arg2P—by modelling two application scenarios, concerning private international law and criminal procedure, respectively.

2. Argumentation and Defeasible Reasoning Technologies: An Overview

In this section, after introducing a running example, we apply some existing implementations of computational argumentation to a legal example.

2.1. Running Example

Let us consider a hypothetical legal case concerning medical malpractice. John was a patient of Doctor Mary and now seeks compensation against her, claiming that she caused him considerable harm, and appeals to a legal rule stating that if a doctor causes harm to a patient, then the doctor has the obligation to compensate the damage, unless it is proven that the doctor was not negligent. We may also add that this rule establishes a presumption of negligence against the doctor: The patient does not need to prove the doctor's lack of care to receive the compensation, but rather, it is the doctor who is tasked with proving non-negligence. There is also a conditioned presumption of non-negligence favouring the doctor: The doctor was careful (i.e., not negligent) in case he follows medical guidelines. Let us assume that expert evidence is provided by the two parties and consider two possible claims by the experts. Mark, the expert witness for John, may claim that there was harm and may also claim that the doctor did not follow the applicable guidelines. Edward,

the expert witness for Mary, may claim that there was no harm and may also claim that the guidelines were followed. In the following, we shall consider different combinations of claims by the experts and see what conclusions they generate in the above-mentioned different approaches.

Two main approaches have been adopted to deal with defeasibility and argumentation in the law: non-monotonic logics and argumentation systems. In the following, we discuss implementations of non-monotonic logics (namely, defeasible logic and ASP) and structured argumentation (namely, DeLP, ABA+, and ASPIC+).

2.2. Defeasible Logic

Defeasible Logic (DL) is a well-known formalism for defeasible reasoning, originally proposed by Nute [21], and later extended in various directions, including deontic logic (see also [22,23]). In our analysis, we used d-Prolog [21] as a defeasible reasoner. Let us assume that Mark claims that there was damage, while Edward claims that the knowledge (the guidelines) was correctly used by doctor Mary. The use case may be encoded as shown in Figure 1 in DL:

```

patient('John').
doctor('Mary').
expert('Mark').
expert('Edward').
say('Mark', harmed(doctor('Mary'), patient('John'))).
say('Edward', careful(doctor('Mary'))).

liable(doctor(D)) := harmed(doctor(D), patient(P)).
neg liable(doctor(D)) := used_correctly(knowledge, doctor(D)).

harmed(doctor(D), patient(P)) := say(X, harmed(doctor(D), patient(P))), expert(X).
used_correctly(knowledge, doctor(D)) := say(X, careful(doctor(D))), expert(X).

```

Figure 1. DL formalism.

Running the query `@liable(doctor('Mary'))` (`@neg liable('Mary')`), we obtain a negative response, since the inferences for `liable` and `not liable` defeat one another.

Adding a priority, as shown in Figure 2, for the rule against liability over the one for liability, we obtain: *yes* for Mary's non-liability.

```

sup_rule(neg liable(doctor(D)) := used_correctly(knowledge, doctor(D)),
liable(doctor(D)) := harmed(doctor(D), patient(P)) ).

```

Figure 2. DL priority.

Assume now that Mark also intervenes on the issue of compliance with the guidelines, claiming that Mary did not follow the guidelines. Surprisingly, running the query `@liable(doctor('Mary'))` again, we obtain a positive result. In fact, the rule on the exclusion of liability would not be triggered, given that the antecedent `used_correctly(knowledge, doctor('Mary'))` could not be established.

This aspect of the functioning of DL is called ambiguity blocking: when two conflicting inferences clash and there is no priority, the inferences cancel one another. Thus, an unsolved conflict between two propositions, rather than producing uncertainty on such propositions and on what follows from them makes both propositions (and their consequences) irrelevant to the reasoning of the system.

We may wonder to what extent this behaviour is appropriate in modelling laws. To this end, let us first introduce the concept of burden of proof. Generally speaking, we can say that burdens of proof distribute dialectical responsibilities between the parties in a dialogue. In other words, when a party has a burden of proof relative to a claim ϕ , then, unless the party provides the kinds of arguments or evidence that is required by that type of burden, the party will lose on the claim ϕ . Losing on the burdened claim means that, for the purpose of the dialectic interaction at stake, it will be assumed that ϕ has not been established, not even as a relevant possibility. So, as regards ambiguity blocking,

on the one hand, it may correspond to an aspect of the burden of proof: If an operative fact that conditions a legal effect is not proved, the effect does not follow. On the other hand, this principle only applies to factual information; it does not apply to conflicts based on competing legal norms (resulting from alternative interpretations or indeterminate priorities). Moreover, when facing alternative factual reconstruction, it may be useful for legal reasoners to know what alternative legal conclusions would be obtained in case one reconstruction would prevail over the other.

Defeasible logic theories that include ambiguity propagation (uncertain inferences may still attack other inferences) have been defined, but we could not find any available applications.

2.3. ASP for Non-Monotonic Reasoning

Answer set programming (ASP) is an approach to logic programming oriented towards difficult (primarily NP-hard) search problems. It is based on the stable model (answer set) semantics and allows for extensions (possible answers) to be computed in a very efficient way. We used Clingo (<https://potassco.org/clingo/run/>, last access 15 October 2021 with its online ASP implementations. We have run the example with the source code, as shown in Figure 3. This input yields no results (no stable models) because of the unsolved contradiction caused by rules one and two.

```
liable(doctor(D)) :- harmed(doctor(D), patient(P)).
not liable(doctor(D)) :- used_correctly(knowledge, doctor(D)).
harmed(doctor(D), patient(P)) :- say(X,harmed(doctor(D), patient(P))), expert(X).
used_correctly(knowledge, doctor(D)) :- say(X,careful(doctor(D))), expert(X).

patient(john).
doctor(mary).
expert(mark).
expert(edward).
say(mark,harmed(doctor(mary), patient(john))).
say(edward,careful(doctor(mary))).
```

Figure 3. ASP formalism.

Note that the standard ASP format (ASP-Core-2), used by systems such as Clingo, does not support the use of preferences over rules. To express that the rule with conclusion *harmed(doctor(D), patient(P))* applies unless the doctor uses correctly the knowledge, we have to introduce a negation by failure *not used_correctly(knowledge, doctor(D))* in the body of that rule. With this addition, Clingo provides an answer according to which there is no liability. Extensions of ASP are being provided that include preferences over rules, but we have not yet tested them.

2.4. Implementation Based on Structured Argumentation

In the following, the running example is discussed in three well-known frameworks for structured argumentation: DeLP, ABA+, and ASPIC+.

2.4.1. DeLP

DeLP is a formalisation of defeasible reasoning in which the results of Defeasible Logic and Argumentation are combined. The formalism is fully introduced in [24].

We used TweetyProject@Web (<http://tweetyproject.org/w/delp/index.html>, last access 15 October 2021), an online DeLP implementation. The use case is encoded using Delp's syntax, as shown in Figure 4:

```

Patient(john).
Doctor(mary).
Expert(mark).
Expert(edward).
Say_harmed(mark, mary, john).
Say_careful(edward, mary).

Liable(D) -< Harmed(D, P).
~Liable(D) -< Used_correctly(knowledge, D).

Harmed(D,P) -< Say_harmed(X,D, P), Doctor(D), Patient(P), Expert(X).
Used_correctly(knowledge, D) -< Say_careful(X,D), Doctor(D), Expert(X).

```

Figure 4. DeLP formalism.

The answers are the same as those given by the DL implementation, though DeLP allows for ambiguity propagation, i.e., it may develop inferences based on conflicted propositions (this corresponds, as we shall see, to the complete semantics in ASPIC). Note the loss of expressiveness in terms of syntax with respect to DL as captured by d-Prolog: DeLP does not allow for function symbols inside predicates.

2.4.2. ASPIC+

ASPIC+ is a popular framework for structured argumentation, exploiting Dung's abstract semantics [25]. ASPIC allows users to choose from different semantics: grounded, preferred, semi-stable, and stable. The preferred semantic is particularly significant for the law, since it shows alternative extensions for unsolved conflicts.

We selected Arg2P [26,27] as our reference implementation. The use case is encoded as shown in Figure 5 with its corresponding generated arguments (Figure 6 and argumentation graph (Figure 7) evaluated under grounded semantics:

```

Premises:
p1 :=> patient(john).
p2 :=> doctor(mary).
p3 :=> expert(mark).
p4 :=> expert(edward).
p5 :=> say(mark,harmed(doctor(mary), patient(john))).
p6 :=> say(edward,careful(doctor(mary))).

Rules:

r1 : harmed(doctor(D), patient(P)) => liable(doctor(D)).
r2 : used_correctly(knowledge, doctor(D)) => -liable(doctor(D)).
r3 : say(X,harmed(doctor(D), patient(P))), expert(X) => harmed(doctor(D), patient(P)).
r4 : say(X,careful(doctor(D))), expert(X) => used_correctly(knowledge, doctor(D)).

```

Figure 5. ASPIC+ formalism.

```

A0 : p2 : doctor(mary)
A1 : p4 : expert(edward)
A2 : p3 : expert(mark)
A3 : p1 : patient(john)
A4 : p6 : say(edward, careful(doctor(mary)))
A5 : p5 : say(mark, harmed(doctor(mary), patient(john)))
A6 : A5,A2,r3 : harmed(doctor(mary), patient(john))
A7 : A4,A1,r4 : used_correctly(knowledge, doctor(mary))
A8 : A6,r1 : liable(doctor(mary))
A9 : A7,r2 : -liable(doctor(mary))

```

Figure 6. ASPIC+ arguments.

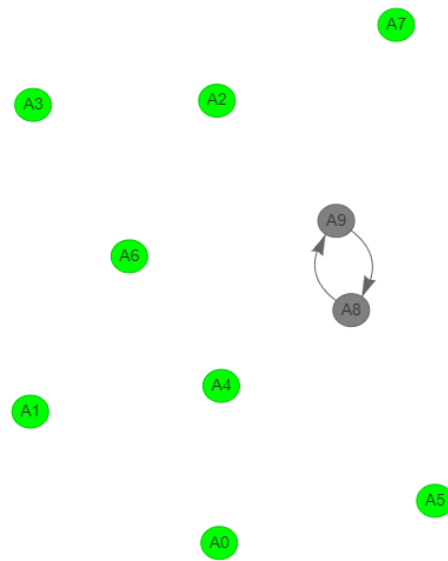


Figure 7. ASPIC+ argumentation graph.

Note that both arguments A8 and A9 are undecided since they defeat one another. This is the right outcome according to the grounded semantics. The results also do not change if we add *-used_correctly(knowledge, doctor(mary))* to our facts. This shows an interesting difference between DL and ASPIC. In DL, an unsolved conflict between two inferences results in such inferences (and the inferences expanding them) becoming irrelevant. In ASPIC, the conflicting arguments can still prevent the other arguments from being included in the final extension.

At the moment, Arg2P allows users to choose between two different semantics: grounded and complete. The complete semantic is particularly significant for the law since it presents users with alternative extensions for unsolved argument conflicts. In other terms, it enables the identification of incompatible arguments that are all legally defensible (not worse than the competing arguments) though not being justified (better than their competitors). This may enable legal analysts to identify uncertainties and consider what would happen if the uncertainties were decided one way rather than the other. For instance, selecting the complete semantics, we obtain the two additional extensions showed in Figure 8.

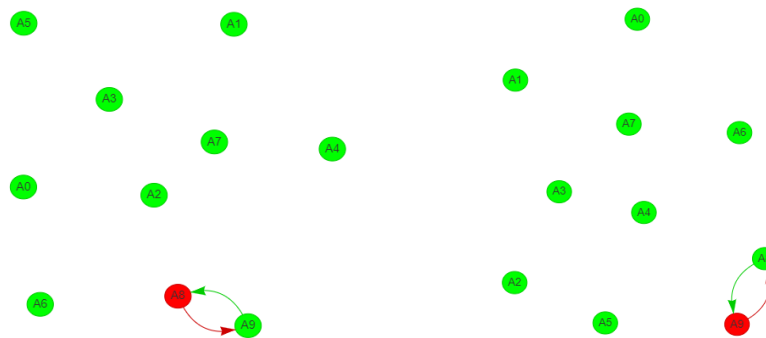


Figure 8. ASPIC+ argumentation graph evaluation under complete semantics. According to the solution on the left, argument A9—the one for Marys’s innocence—is accepted. Conversely, according to to solution on the right, Mary should be considered liable (A8).

The assessment changes if we add priorities between rules. In this regard, Arg2P allows users to choose between multiple principles. Let us consider the last link principle according to which an argument prevails if its top rule is stronger than the top rule of the opposing argument. In our example, if we add a preference for rule r_2 over rule r_1 ($sup(r_2, r_1)$), we obtain that A9 is now accepted (justified), while A8 is rejected.

2.4.3. ABA+

In ABA+, arguments are sets of assumptions used to infer a conclusion. We used the online implementation ABAPlus <http://www-abaplus.doc.ic.ac.uk>, last access on 17 October 2021. The example case is encoded as shown in Figure 9, while Figure 10 shows the resulting assumption graph:

```

Assumptions:
myAsm(liable).
myAsm(neg_liable).

contrary(liable , neg_liable_mary).
contrary(neg_liable , liable_mary).

Facts:
myRule(patient_john , []).
myRule(doctor_mary , []).
myRule(expert_mark , []).
myRule(expert_edward , []).
myRule(say_harmed_mark_mary_john , []).
myRule(say_careful_edward_mary , []).

Rules:
myRule(harmed_mary_john , [say_harmed_mark_mary_john , doctor_mary , patient_john , expert_mark]).
myRule(used_correctly_manual_mary , [say_careful_edward_mary , doctor_mary , expert_edward]).
myRule(liable_mary , [harmed_mary_john , doctor_mary , liable ]).
myRule(neg_liable_mary , [doctor_mary , used_correctly_manual_mary , neg_liable]).
    
```

Figure 9. ABAPlus formalism.

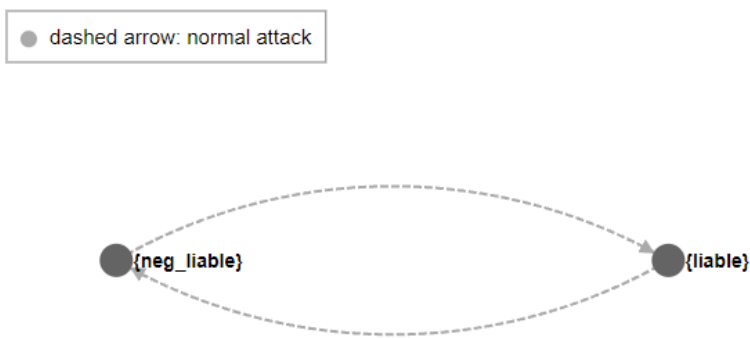


Figure 10. ABAPlus assumption graph.

In ABAPlus, each rule has to be ground: General rules containing variables cannot be expressed. This is due to the fact that ABAPlus uses a semantics-preserving mapping from ABAPlus to AA and employs ASPARTIX <https://www.dbai.tuwien.ac.at/proj/argumentation/systempage/>, last access on 19 October 2021 for determining extensions. Moreover, ABA+ does not deal with preferences over rules; it only has preferences over assumptions. Preferences over rules can be simulated by, e.g., adding dummy assumptions to the rules and expressing (contraries and) preferences accordingly. An additional burden to the notation is represented by the fact that assumptions need to have contraries declared.

3. Arg2P: An Inclusive Approach to Model Diversity

The examples above have shown that there is a fundamental overlap between all the presented implementations. All of them enable the use of defeasible inference. This is obtained by providing for defeasible rules (or through the use of assumption predicates as in ABA). Preferences among rules are also supported by all the considered approaches (in ABA, preferences concern assumptions).

Some other features are only present in certain implementations or are provided with different functionalities.

Only some implementations allow for combinations of strict and defeasible inference, through defeasible rules (as in ASPIC+ and DL) or through other mechanisms (by using or not using assumptions in ABA). This feature may be useful when legal knowledge, typically based on defeasible rules, has to be combined with other kinds of knowledge (e.g., non-defesible laws from mathematics or empirical sciences).

Defeasible inferences take place in two different models to which we refer to as ambiguity propagation and ambiguity blocking. In the first model (used in ABA, ASPIC+, and ASP), uncertainty on an argument does not prevent that argument from successfully challenging other arguments. In the second model (natively used in DL and DeLP), uncertain arguments are completely ineffective. Both modes of inference may be meaningful in legal reasoning, e.g., ambiguity blocking may be needed when dealing with burdens of proof (burdens of persuasion), while ambiguity propagation may be appropriate when no such burden is at stake.

Some approaches (such as DL) provide features for representing deontic concepts and reasoning with them, but we cannot find such features in the available implementations. Similarly, some approaches, such as DL and ASPIC+, provide features for capturing reasoning schemes, but this aspect, too, was not reflected in the available implementations. Deontic reasoning may be useful when violations of mandatory norms have to be taken into account, but may not be needed when constitutive norms establishing requirements and entitlements are being processed.

Contrarily to other approaches, defeaters in DL and DeLP can be expressed via explicit rules. In DL and in argumentation-based approaches, different types of attacks are distinguished (e.g., undercutting, rebutting, or undermining), while DeLP defines a single general notion of attack, which accounts for all the possible situations in which two arguments are considered to be conflicting.

If an application has to use the same general rules for multiple cases, the possibility of using open (non-grounded) rules in the knowledge base, and of using different instances of the same predicates in different rules is a key advantage. This is a feature that we could find in all of the described systems, except for ABA+.

When a system has to deal with a high number of uncertain conflicts, which it is not possible to decide in advance, we think that the ability to rely not only on skeptical but also on credulous reasoning may be important. This is a feature that we could find in the argumentation-based approaches, such as DeLP, ASPIC+, and ABA+. The ability of ASPIC+ to switch between grounded and preferred semantics was particularly user friendly.

When a system has to address complex issues of legal reasoning, and full explicability is required (which also include explanations of what are the alternatives at stake and why one was chosen over the other), we think that the ability to provide a picture of existing arguments and of the relations between them and a semantic explaining what arguments should or could be finally endorsed may become a decisive feature. This is a feature we could find in ASPIC+ and ABA+, though we found that the ABA+ way of modelling knowledge and arguments much less intuitive for a legal user. An explicit representation of the dialectics of competing arguments, however, may not be so relevant when a system has to apply a piece of legislation, with regard to which relevant priorities could be decided in advance. In such cases, the efficiency of a reasoning tool such as DL or ASP (or just Prolog) can be the decisive factor.

All different features and functionalities just presented may be useful or not, depending on the legal domain being addressed and on the purposes of the project at issue. This consideration has led to the development of Arg2P, an integrated framework for modelling legal knowledge which can be parameterised to provide knowledge representation and inference features that are appropriate to specific application domains. In the following, we demonstrate how it can be used in different application contexts, to capture legal knowledge and reasoning in a way that fits lawyers' intuitions and expectations.

3.1. The Technology: Arg2P

Arg2P (We remind the reader to visit the Arg2P homepage: <http://arg2p.apice.unibo.it> for a complete overview of the tool, last access on 15 October 2021) [26,27] is a lightweight Prolog-based implementation for structured modular argumentation [28,29] that combines modular logic programming and legal reasoning. It allows also for some patterns of legal reasoning to be captured—such as the burden of persuasion [30,31] and deontic concepts [32] (with an explicit formalisation of permissions and obligations).

Modular argumentation enables reasoning with rules and interpretations of multiple domains (e.g., different legal systems) and enables the design of knowledge organised in distinct and separate modules that can “call” one another. In particular, a knowledge module—which may represent a legal system or parts of it—can refer to another module for specific issues by querying it directly.

Modules in Arg2P are identified by distinct theories (Prolog files) and can be called and executed exploiting the predicate `module_call(+Modules, :Query)`, where `Modules` is a list of theories (Prolog modules) that need to be loaded to answer the query, and `Query` is the query that must be evaluated. In particular, the predicate: (i) creates a new environment that contains *only* the required modules’ data, (ii), executes the query in the newly created environment, (iii) and feeds the result to the caller—note that the original caller environment is not altered by the procedure.

Both strict and defeasible rules can be defined according to two equivalent syntaxes: a Prolog-like syntax, where conclusions precede premises, or an ASPIC⁺-like syntax [16], where premises precede conclusions in the rule syntax.

Accordingly, strict rules can be represented both in the Prolog-like syntax form as $c : -p_1, \dots, p_n$ or in the equivalent ASPIC-like syntax as $label : p_1, \dots, p_n - > c$ where $label$ is the identifier of a rule and conclusion c , and premises p_1, \dots, p_n are Prolog terms. Defeasible rules can be represented by using the operator $=>$ (*defeasibly entails*) instead of $- >$ (*strict entails*).

Two distinct negations can be used in rules: *strong negation* available via the $-$ operator—which captures the notion of definite false—and *weak negation*, i.e., negation as failure in rule premises through the $/+$ operator (Prolog-like syntax) or \sim operator (ASPIC-like syntax).

Different semantics can be computed in the framework specifying the value of the *argumentLabellingMode*: The currently available semantics are *grounded*, *complete*, *bp_grounded*, *_partial*, and *bp_grounded_complete* (details in [33]).

3.2. Case Studies in the Legal Domain: ASPIC+ Approach via Arg2P

To demonstrate how Arg2P can be used in different legal contexts, we describe two case studies focused on private international law (PIL) and criminal justice, respectively.

The first one, focused on PIL, has been developed in the context of the InterLex project (Further project information is available at <https://interlex-portal.eu/>, last access on 15 October 2021), aimed at assisting legal professionals in the identification of competent courts and applicable laws in disputes regarding Internet-related PIL. A structured model of this legal domain can help to ensure that different national legal frameworks and regulations are in line with the EU *acquis* and legislation. As the resulting system is a multilevel one, a careful analysis is needed to solve possible conflicts between legal sources, e.g., by granting priority to higher-ranking norms (principle of *lex superior*), as well as by considering their temporal effect and territorial scope. In particular, the Interlex knowledge base includes a logic translation of the Brussels Regulation, the Rome I and Rome II Regulations, the GDPR, and a set of relevant national laws.

In this context, argumentation can be used to provide users with:

- information on jurisdiction and applicable law;
- a visual representation of conflicts between different legal systems;

- a feature supporting personal preferences, e.g., whenever multiple courts are competent to judge a case, users can express a preference either for their place of residence or their place of business.

The second case study focuses on criminal justice. It has been developed in the context of the CrossJustice project (The webpage is available at <https://www.crossjustice.eu/>, last access on 15 October 2021), dealing with defendants' rights in criminal matters. In particular, we developed a decision support system, freely accessible by legal professionals and citizens, to assess (a) criminal procedural rights granted to accused and suspects, as well as (b) the level of harmonisation between national rules and relevant EU Directives. Indeed, such Directives are not directly applicable by Member States, since they need to be transposed in national criminal laws. This may lead to discrepancies, often originating from (a) differences in the used terminology, possibly referring to diverse legal concepts, and (b) the omission or addition of certain legal requirements by national laws. For instance, under the EU Directive 2013/48, any accused has the right to consult a lawyer, while under the Dutch transposition, the accused must also be deprived of liberty.

3.3. Application in the Private International Law Domain

As an example in the PIL domain, let us consider the following hypothetical case. Mr. Nauru bought some furniture for his apartment in Milan from a website owned by Mr. Milieu. They are both French nationals. However, while Mr. Nauru resides in Paris, Mr. Milieu is domiciled in Rome. After receiving the expected payment, Mr. Milieu did not ship the furniture. As a consequence, Mr. Nauru decided to sue the seller.

In the following, we present how the relevant EU and national laws have been modelled to determine the competent court.

Figure 11 illustrates the applicable articles of the Brussels Regulation (Regulation (EU) No 1215/2012 of the European Parliament and of the Council of 12 December 2012 on jurisdiction and the recognition and enforcement of judgments in civil and commercial matters (recast)). The first is article 4, which states that *persons domiciled in a Member State shall, whatever their nationality, be sued in the courts of that Member State*. The second is article 7, paragraph 1, letter a, which states that *a person domiciled in a Member State may be sued in another Member State in the courts for the place of performance of the obligation in question*.

```
% Subject to this Regulation, persons domiciled in a Member
% State shall, whatever their nationality, be sued in the courts of
% that Member State.

hasJurisdiction(art4, Country, Court, ClaimId, brusselsRegulation):-
    user_fact(personRole(PersonId, ClaimId, defendant)),
    user_fact(personDomicile(PersonId, Country, Court)),
    user_fact(memberState(Country)).

% A person domiciled in a Member State may be sued in another Member State:
% (1) (a) in matters relating to a contract, in the courts for the place of
% performance of the obligation in question;

hasJurisdiction(art7_1, Country, Court, ClaimId, brusselsRegulation) :-
    user_fact(claimObject(ClaimId, contract, ContractId)),
    user_fact(placeOfPerformance(Country, Court)).
```

Figure 11. Brussels Regulation Prolog transposition.

Figure 12 illustrates article 14 of the French Civil Code, which states that *even if not residing in France, a person may be cited before French courts for the performance of obligations contracted by him in a foreign country towards French persons*.

```
% An alien, even if not residing in France, may be cited before French courts for the performance of obligations
% contracted by him in France with a French person; he may be called before the courts of France for obligations
% contracted by him in a foreign country towards French persons.

hasJurisdiction(art14, france, Court, ClaimId, frenchCivilCode) :-
    user_fact(claimObject(ClaimId, contract, ContractId)),
    user_fact(personRole(PersonId, plaintiff)),
    user_fact(personNationality(PersonId, french, Court)).
```

Figure 12. French Civil Code Prolog transposition.

Figure 13 shows the list of facts to be added as input for the evaluation of the specific case. The object of the claim is a matter of contract, which we have indicated with the identification of 'contract_1'. The place of performance of the obligation in question is Italy, Milan. The parties, i.e., Mr. Nauru (the plaintiff) and Mr. Milieu (the defendant) are both French nationals, and the latter is domiciled in Italy.

```
% Facts
user_fact(claimObject(claim_1, contract, contract_1)).
user_fact(placeOfPerformance(italy, milan)).
user_fact(personRole(nauru, plaintiff)).
user_fact(personNationality(nauru, french, paris)).
user_fact(personRole(milieu, claim_1, defendant)).
user_fact(personDomicile(milieu, italy, rome)).
```

Figure 13. Facts Prolog transposition.

Figure 14 illustrates the two possible conflicts that have been highlighted for this example. The first states that whenever the system finds two alternative jurisdictions in analysing the same case, and they are obtained from different legal sources, it shall provide a preferred solution based on the hierarchy of the legal source of origin. In this example, we have the Brussels Regulation and the French Civil Code. Since the former has direct effect in the Member State's legal systems, it defeats the latter. This has been made explicit by giving priority to all rules found in the Brussels module over those found in the French one. As for the second conflict, it is used to show the user the different available jurisdictions (if they are all applicable). The user can then decide which solution is best according to his knowledge of the case. Since one of the characteristics of private international law consists in the parties having several possible jurisdictions to choose from the so called 'forum shopping', this conflict helps parties realise where the choice of jurisdiction comes from and enables the use of priorities to select preferred rules over others.

To reach this goal, rules a1 and a2 have been added to set a preferred jurisdiction based on certain facts, i.e., the domicile of the subject, and the place of performance. In a general case, these preferences could be added by the user.

For this focus case, let us imagine that the user has decided that all articles concerning the place of performance of the obligation shall be preferred over the ones that relate to the domicile of the defendant, as it is in his interest to have the location of the court as close to his new house as possible. This is displayed in our system through the following rule `sup(a2, a1)`. As EU Regulations become part of a member state's national law without the need for any domestically implemented legislation, most conflicts are always solved in favour of the supranational source (This is declared in the code as the second preference, stating that all rules originating from the Brussels Regulation shall be preferred over the ones in the French Civil Code). As preferences are easily modifiable and can be rearranged based on the circumstances of the case, such a method is optimal to give the user more information regarding the system's result and the different choices which are presented. Legal professionals, who might have a deeper understanding of the national legal systems, may lack knowledge of the intricacies of private internal law. This feature may be particularly useful to create arguments advantageous to their preferred jurisdiction, which applies the more favourable laws.

```

modulesPath('path/to/the/modules/folder').

generate :
  module(Module),
  prolog(call_module([Module, 'facts'],
    with_facts_and_length(hasJurisdiction(Art, Country, Court, ClaimId, Code), Facts, _)))
    -> jurisdiction(Art, Country, Court, ClaimId, Code, Facts).

module1 : [] => module('brussels').
module2 : [] => module('frenchCivilCode').

a1 : jurisdiction(Art, Country, Court, ClaimId, Code, Facts), prolog(member(personDomicile(_, _, _), Facts)) =>
  preferred_jurisdiction(domicile, Art, Country, Court, ClaimId, Code).
a2 : jurisdiction(Art, Country, Court, ClaimId, Code, Facts), prolog(member(placeOfPerformance(_, _), Facts)) =>
  preferred_jurisdiction(place_of_performance, Art, Country, Court, ClaimId, Code).

conflict(
  [jurisdiction(Art1, Country1, Court1, ClaimId, Code1, Facts1)],
  [jurisdiction(Art2, Country2, Court2, ClaimId, Code2, Facts2)]
) :- Country1 \= Country2.

conflict(
  [preferred_jurisdiction(Object1, Art1, Country1, Court1, ClaimId, Code1)],
  [preferred_jurisdiction(Object2, Art2, Country2, Court2, ClaimId, Code2)]
) :- Object1 \= Object2.

sup(a2, a1).
sup(module1, module2).

```

Figure 14. Conflict Prolog transposition.

3.4. Discussion

Figure 15 shows the generated arguments on the left and the visualisation of the results of the framework evaluation according to grounded semantic on the right, on the basis of the facts added as input.

```

A0 : module1 => module(brussels)
A1 : module2 => module(frenchCivilCode)
A2 : A1,generate => jurisdiction(art14, france, paris, claim_1,
  frenchCivilCode, [personNationality(nauru, french, paris), personRole(
  nauru, plaintiff), claimObject(claim_1, contract, contract_1)])
A3 : A0,generate => jurisdiction(art4, italy, rome, claim_1,
  brusselsRegulation, [memberState(italy), personDomicile(milieu, italy,
  rome), personRole(milieu, claim_1, defendant)])
A4 : A0,generate => jurisdiction(art7_1, italy, milan, claim_1,
  brusselsRegulation, [placeOfPerformance(italy, milan), claimObject(
  claim_1, contract, contract_1)])
A5 : A3,a1 => preferred_jurisdiction(domicile, art4, italy, rome, claim_1,
  brusselsRegulation)
A6 : A4,a2 => preferred_jurisdiction(place_of_performance, art7_1, italy,
  milan, claim_1, brusselsRegulation)

```

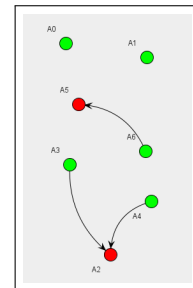


Figure 15. PIL example.

First of all, the solution presented gives a choice between the jurisdiction of France, according to article 14 of the French Civil Code (Argument A2), as well Italy, Rome, according to article 4 of the Brussels Regulation (Argument A3) and Italy, Milan, according to article 7(1) of the same (Argument A4). Furthermore, article 4 is based on the domicile of the person (Argument A5), while article 7(1) is based on the place of performance of the obligation (Argument A6)

Confronting these results with the graph on the right, it is shown that the article of the French Civil Code is defeated by the articles originating from the Brussels Regulation, giving priority to the European legal source over the national one. Furthermore, the user can visualise how the preference he has added to the system results in article 7(1) defeating article 4, showing the jurisdiction based on the place of performance is preferred over that based on the domicile of the defendant.

This feature may assist the user in interacting with the rule base and in visualising a structure of arguments that explains and describes the situation, even working outside of the strict legal terms. Depending on the specific goal the user is trying to solve, this aspect of argumentation allows for the creation of personal relations between arguments, and it can be changed to include different elements, such as the Country of Jurisdiction or the number of conditions involved.

3.5. Application in Criminal Justice

Let us consider Nino, an Italian citizen who has been placed into temporary detention, being formally accused of a crime committed in Holland, and is about to be questioned by the judicial authority. The reasoning system may be used to assess which rights are granted to Nino. The example will only take into account the right to access and communicate with a lawyer and the right not to incriminate oneself. The role of argumentation in this case is not only to provide an answer concerning the subject's rights but also to verify the conformity between the rights granted by national law and those granted by the European Directives, as well as between the respective reasons for granting such rights.

```

conflict(
  [right_with_facts(X, Article, PersonId, Right, Matter, F)],
  [right_with_facts(X1, Article1, PersonId, Right, Matter, F1)]
) :- X \= X1, sort(F, SF), sort(F1, SFF), SFF \= SF.

conflict(
  [right_with_facts(X, Article, PersonId, Right, Matter, F)],
  [right_with_facts(X1, Article1, PersonId, Right, Matter1, F1)]
) :- X \= X1, Matter \= Matter1.

% r1
right_with_facts(X, Y, PersonId, Z, U, Facts) :-
  has_right(X, Y, PersonId, Z, U),
  fetch_argument_facts(has_right(X, Y, PersonId, Z, U), Facts).

% r2
right_with_facts(X, Y, PersonId, Z, U, Facts) :-
  auxiliary_right(X, Y, PersonId, Z, U),
  fetch_argument_facts(auxiliary_right(X, Y, PersonId, Z, U), Facts).

```

Figure 16. Conflicts added to the Prolog source.

The example highlights a specific type of conflict, namely, the attack–defeat relation between articles of the European law and articles of national legislation. This example deals in particular with European Directives, where there is more possibility for conflicts with national legal systems, given the freedom granted to Member States in form and methods for transposing directives in national law. The first conflict modelled in the example, as shown in Figure 16, states that whenever the same right and matter (In modelling the rules, the word “matter” has been used to identify the object to which a right applies, e.g., the specific phase of the proceedings) is returned but the considered facts (the requisites) are different, a conflict shall arise. Another conflict may also arise whenever the system returns two results granting the same right, albeit with a different matter. The system then selects the set of relevant facts used in evaluating the result under national law, through the custom predicate `right_with_facts`. The selected facts are then compared to those used for the evaluation under the European Directive. Thus, under national and European law, the sets of facts may either be different or, even if identical, be phrased in such way as to lead to different interpretations. Even though interpretative issues are a matter for judicial rulings, it is interesting to highlight such discrepancies and the potential differing legal effects.

Figure 17 is the representation of Article 3(2) of Directive 2013/48 (Rule r3), which states that suspects or accused persons shall have access to a lawyer without undue delay before they are questioned by the police or by another law enforcement or judicial authority. We have represented this right as ‘`right_to_access_lawyer`’, with the matter beginning ‘`before_questioning`’. The only requirement is for the person to be officially accused.

Furthermore, Article 3(3) of the same Directive (Rule r5) grants the accused the right to meet in private and communicate with the lawyer representing them. In the system, this rule is defined as an “auxiliary right”, since it is dependent on the granting of a main right. The link between a main right and an auxiliary right can either be a temporal one, so that a right exists only after the main one has been applied (for example, the right to remedy to a previous decision), or a subjective one, which implies that the defendant has particular needs, must explicitly submit a request or must act in order to have a right recognised (for example, the right to have the costs of the interpreter be paid by the state). In our system, is linked to the “main right” by an additional predicate (Rule r4). This right is always granted to the person unless there is an urgent need to avert serious adverse consequences for the

life, liberty, or physical integrity of a person. Article 3(6) (Rule r6) is thus represented as an auxiliary right and is considered an exception to Article 3(3).

```

% Directive 2013/48
% r3
has_right(dir, art3_2_a, PersonId, right_to_access_lawyer, before_questioning) :-
    user_fact(person_status(PersonId, accused)).

% r4
auxiliary_right_scope(art3_3_a, [art3, art3_2_a, art3_2_b, art3_2_c]).

% r5
auxiliary_right(dir, art3_3_a, PersonId, right_to_communicate, private_communication) :-
    auxiliary_right_scope(art3_3_a, X),
    member(Y, X),
    has_right(dir, Y, PersonId, Right, Matter),
    \+ exception(auxiliary_right(dir, art3_3_a, PersonId, right_to_communicate, private_communication), _).

% r6
exception(auxiliary_right(dir, art3_3_a, PersonId, right_to_communicate, private_communication), art3_6_a) :-
    user_fact(person_danger(PersonId, life)).

% Directive 2016/343
% r7
has_right(dir, art7_2, PersonId, right_to_not_incriminate_themselves, proceeding) :-
    user_fact(person_status(PersonId, accused)),
    \+ exception(has_right(art7_2, PersonId, right_to_not_incriminate_themselves, proceeding), _).

```

Figure 17. Directive transposition in Prolog.

Finally, the representation of Article 7(2) of Directive 2016/343 (Rule r7) expresses the fact that Member States shall ensure that suspects and accused persons have the right not to incriminate themselves. This has been modelled as ‘right_to_not_incriminate_themselves’, while the matter is ‘proceedings’ as it is not limited to a single phase of the trial. The only condition for the recognition of such a right is the fact that the person has been officially accused.

Figure 18 shows the representation of the articles in the Dutch code of criminal procedure (hereinafter CCP), starting from Article 28b (Rule r8) which gives the accused the right to be assisted by a lawyer only after he has been deprived of liberty. Under Dutch law, the authorities are required to take active steps to make it possible for the suspect to have access to a lawyer only following their arrest. While the Dutch legislator recognises that the right of access to a lawyer in criminal proceedings also applies to the suspect not deprived of their liberty, it did not consider it necessary to expressly provide for this right in the CCP. Thus, while the Code of Criminal Procedure expressly provides for the right of the suspect already deprived of their liberty to consult a lawyer prior to questioning, it does not do so for the suspect whose liberty was not affected, since they are still in a position to autonomously arrange a consultation with a lawyer.

Article 45(1) of the Dutch CCP, represented in Rule r10, also grants the accused the right to meet in private and communicate with a lawyer. While Rule r10 would be by itself a perfect transposition of the corresponding provision in the directive, since it is an auxiliary right linked to the main article of the Dutch CCP (represented in Rule f8) it suffers from the same issue described previously.

The Dutch legislator has implemented this part of the Directive in different legal sources, such as the Custodial Institutions (Framework) Act. Article 36 (Rule r11) of such Act recognises the prisoner the right to send and receive letters by post, including correspondence with his lawyer, while Article 41 (Rule r12) recognises the same right to the accused. Although the right is the same as the one in Article 3(3) of the Directive, its matter is different (letter vs. private communication).

Rule r13 is the representation of Article 29(1) of the Dutch CCP, which states that, in all cases where a person is being questioned as a suspect or defendant, the judge or officer conducting the questioning shall refrain from any act aimed at obtaining a statement which cannot be said to have been freely given. This is the transposition of Article 7(2) of the Directive. However, while the directive provides a broader scope, i.e., the right to not incriminate oneself in the proceedings, the Dutch implementation limits the applicability of the right to the questioning of the person.

```

% Dutch Code of Criminal Procedure
% r8
has_right(dutch, art28b, PersonId, right_to_access_lawyer, trial) :=
  user_fact(person_status(PersonId, accused)),
  user_fact(person_status(PersonId, deprived_of_liberty)).

% r9
auxiliary_right_scope(art45_1, [art28b]).

% r10
auxiliary_right(dutch, art45_1, PersonId, right_to_communicate, private_communication) :=
  auxiliary_right_scope(art45_1, X),
  member(Y, X),
  has_right(dutch, Y, PersonId, Right, Matter).

% Custodial Institutions (Framework) Act
% r11
has_right(dutch, art36, PersonId, right_to_communicate, letters) :=
  user_fact(person_status(PersonId, prisoner)).

% r12
has_right(dutch, art41, PersonId, right_to_communicate, letters) :=
  user_fact(person_status(PersonId, accused)),
  \+ exception(has_right(dutch, art41, PersonId, right_to_communicate, letters), art41_4).

% Dutch Code of Criminal Procedure
% r13
has_right(dutch, art29_1, PersonId, right_to_not_incriminate_themselves, questioning) :=
  user_fact(person_status(PersonId, accused)).

```

Figure 18. Dutch law transposition in Prolog.

The system is built to highlight the possible inconsistencies between legal sources and to give an account via argumentation. In the example above, those rules that represent national laws are set as defeasible, while those from EU Directives are strict, thus having a precedence over national law. Therefore, when a rule representing a Dutch norm is attacked by a rule representing a European norm, the latter shall defeat the former. Our functions expressing conflicts only compare rules from different sources, thus the European norms shall always prevail over national ones. European rules can only be defeated by exceptions found in other EU rules, as we will observe in the following example.

Figure 19 shows the facts used to evaluate this example case, in particular the fact that Nino has been accused and that he is a prisoner and has thus been deprived of liberty. Furthermore, we state that there is an urgent danger of serious adverse consequences for the life Nino.

```

% Dutch Code of Criminal Procedure
% r8
has_right(dutch, art28b, PersonId, right_to_access_lawyer, trial) :=
  user_fact(person_status(PersonId, accused)),
  user_fact(person_status(PersonId, deprived_of_liberty)).

% r9
auxiliary_right_scope(art45_1, [art28b]).

% r10
auxiliary_right(dutch, art45_1, PersonId, right_to_communicate, private_communication) :=
  auxiliary_right_scope(art45_1, X),
  member(Y, X),
  has_right(dutch, Y, PersonId, Right, Matter).

% Custodial Institutions (Framework) Act
% r11
has_right(dutch, art36, PersonId, right_to_communicate, letters) :=
  user_fact(person_status(PersonId, prisoner)).

% r12
has_right(dutch, art41, PersonId, right_to_communicate, letters) :=
  user_fact(person_status(PersonId, accused)),
  \+ exception(has_right(dutch, art41, PersonId, right_to_communicate, letters), art41_4).

% Dutch Code of Criminal Procedure
% r13
has_right(dutch, art29_1, PersonId, right_to_not_incriminate_themselves, questioning) :=
  user_fact(person_status(PersonId, accused)).

```

Figure 19. Facts in Prolog.

3.6. Discussion

Figure 20 shows the arguments generated by Arg2P while Figure 21 shows the result of the framework evaluation. Since the Dutch legislator only grants the right during the questioning (Argument A14), instead of the entire proceedings as stated by the European

Directive (Argument A13), the latter defeats the former, indicating that the matter (i.e., the temporal validity) of the right is different.

Looking at the relation between arguments A12 and A17 the graph in Figure ?? shows also that the right to have access to a lawyer is granted. However, while the Directive dictates the right to be granted at the earliest possible time, the Dutch legislator has only recognised the right to those deprived of liberty. In this example, our system still displays the discrepancy that rises from the different conditions used to reach the same conclusion.

```

A0 : r4 => auxiliary_right_scope(art3_3_a, [art3, art3_2_a, art3_2_b, art3_2_c])
A1 : r9 => auxiliary_right_scope(art45_1, [art28b])
A2 : f1 => user_fact(person_danger(nino, life))
A3 : f3 => user_fact(person_status(nino, accused))
A4 : f4 => user_fact(person_status(nino, deprived_of_liberty))
A5 : f2 => user_fact(person_status(nino, prisoner))
A6 : A2,r6 => exception(auxiliary_right(dir, art3_3_a, nino, right_to_communicate, private_communication), art3_6_a)
A7 : A3,r3 => has_right(dir, art3_2_a, nino, right_to_access_lawyer, before_questioning)
A8 : A3,r7 => has_right(dir, art7_2, nino, right_to_not_incriminate_themselves, proceeding)
A9 : A3,r13 => has_right(dutch, art29_1, nino, right_to_not_incriminate_themselves, questioning)
A10 : A5,r11 => has_right(dutch, art36, nino, right_to_communicate, letters)
A11 : A3,r12 => has_right(dutch, art41, nino, right_to_communicate, letters)
A12 : A7,r1 => right_with_facts(dir, art3_2_a, nino, right_to_access_lawyer, before_questioning, [user_fact(
person_status(nino, accused))])
A13 : A8,r1 => right_with_facts(dir, art7_2, nino, right_to_not_incriminate_themselves, proceeding, [user_fact(
person_status(nino, accused))])
A14 : A9,r1 => right_with_facts(dutch, art29_1, nino, right_to_not_incriminate_themselves, questioning, [user_fact(
person_status(nino, accused))])
A15 : A10,r1 => right_with_facts(dutch, art36, nino, right_to_communicate, letters, [user_fact(person_status(nino,
prisoner))])
A16 : A11,r1 => right_with_facts(dutch, art41, nino, right_to_communicate, letters, [user_fact(person_status(nino,
accused))])
A17 : A20,r1 => right_with_facts(dutch, art28b, nino, right_to_access_lawyer, trial, [user_fact(person_status(nino,
accused)), user_fact(person_status(nino, deprived_of_liberty))])
A18 : A21,r2 => right_with_facts(dir, art3_3_a, nino, right_to_communicate, private_communication, [])
A19 : A22,r2 => right_with_facts(dir, art45_1, nino, right_to_communicate, private_communication, [])
A20 : A3,A4,r8 => has_right(dutch, art28b, nino, right_to_access_lawyer, trial)
A21 : A0,A7,r5 => auxiliary_right(dir, art3_3_a, nino, right_to_communicate, private_communication)
A22 : A1,A20,r10 => auxiliary_right(dutch, art45_1, nino, right_to_communicate, private_communication)
    
```

Figure 20. Arguments of the CrossJustice example.

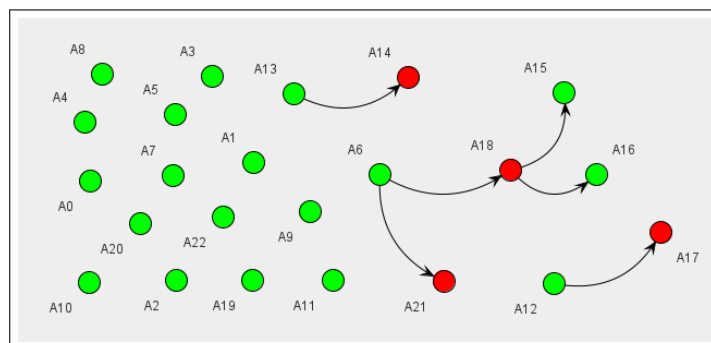


Figure 21. Graphical result of the CrossJustice example.

Lastly, we shall describe the relation that, starting from Argument A6 leads to arguments A21, A18, A15, and A16. The exception in article 3(6) attacks and defeats the right to communicate in private with the lawyer. In this case, a European norm has been defeated, and its attacks towards the relevant Dutch transpositions will not result in the same priority/hierarchy described above. Arguments A15 and A16 are being successfully applied even though neither is a direct transposition of the Directive due to the fact that the attacking article has already been defeated by another rule.

4. Conclusions and Way Forward

Our analysis has shown a strong convergence between the different systems for defeasible reasoning, while they also offer different solutions that may cater for the needs of different legal applications. We think that the times are mature for developing an integrated and modular approach, such as Arg2P, which implements different approaches and semantics in a single working environment.

However, in the future development of argumentation tools for the law, considerations pertaining to the soundness and scope of the underlying logic and reasoning models are not sufficient; they have to be combined with considerations pertaining to the robustness

and usability of argumentation tools. Many improvements are indeed needed to make existing tools really user-friendly and effective (including when deployed in a distributed environment) [34], as well as documented and easily downloadable/deployable. In our future work on Arg2P, we intend to pay due attention to the latter aspects as well, so that our system may support effective applications of defeasible argumentation in legal domains [35].

Author Contributions: This paper reflects ideas the authors have worked out in common, to present and expand the results of previous research projects where all authors contributed, deploying their expertise from different areas. For this reason, all sections in the paper shall be attributed as a joint work to all authors, with their contributions merged into a unitary whole. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the H2020 ERC Project “CompuLaw” (G.A. 833647), by the European Union’s Justice Programme (G.A. 847346) for the project “Knowledge, Advisory and Capacity Building Information Tool for Criminal Procedural Rights in Judicial Cooperation”, and by the European Union’s Justice Programme (G.A. 800839) for the project “InterLex: Advisory and Training System for Internet-related private International Law”.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

- Allen, C.C. State of Mind in Civil Cases. In *Washington University Law Review*; Washington University School of Law: St. Louis, MO, USA, 1957.
- Sergot, M.J.; Sadri, F.; Kowalski, R.A.; Kriwaczek, F.; Hammond, P.; Cory, H.T. The British Nationality Act as a logic program. *Commun. ACM* **1986**, *29*, 370–386. [[CrossRef](#)]
- Bench-Capon, T.J.; Coenen, F.P. Isomorphism and legal knowledge based systems. *Artif. Intell. Law* **1992**, *1*, 65–86. [[CrossRef](#)]
- Governatori, G.; Rotolo, A.; Sartor, G. Temporalised normative positions in defeasible logic. In Proceedings of the Conference 10th International Conference on Artificial Intelligence and Law, Bologna, Italy, 6–11 June 2005; pp. 25–34.
- Ashley, K.D. Reasoning with cases and hypotheticals in HYPO. *Int. J. Man-Mach. Stud.* **1991**, *34*, 753–796. [[CrossRef](#)]
- Hage, J. Law and defeasibility. In *Studies in Legal Logic*; Springer: Cham, Switzerland, 2005; pp. 7–32.
- Prakken, H.; Sartor, G. Law and logic: A review from an argumentation perspective. *Artif. Intell.* **2015**, *227*, 214–245. [[CrossRef](#)]
- Calegari, R.; Ciatto, G.; Omicini, A. On the integration of symbolic and sub-symbolic techniques for XAI: A survey. *Intell. Artif.* **2020**, *14*, 7–32. [[CrossRef](#)]
- Zhang, J.; Chen, B.; Zhang, L.; Ke, X.; Ding, H. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open* **2021**, *2*, 14–35. [[CrossRef](#)]
- Bianchi, F.; Palmonari, M.; Hitzler, P.; Serafini, L. Complementing logical reasoning with sub-symbolic commonsense. In Proceedings of the International Joint Conference on Rules and Reasoning, Bolzano, Italy, 16–19 September 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 161–170.
- Prakken, H.; Sartor, G. Argument-based extended logic programming with defeasible priorities. *J. Appl. Non-Class. Logics* **1997**, *7*, 25–75. [[CrossRef](#)]
- Gordon, T.F.; Walton, D. Legal Reasoning with Argumentation Schemes. In Proceedings of the Conference Conference: The 12th International Conference on Artificial Intelligence and Law, Barcelona, Spain, 8–12 June 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 137–146. [[CrossRef](#)]
- Hage, J.C. *Reasoning with Rules*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 78–129.
- Bench-Capon, T.J.M.; Dunne, P.E. Argumentation in artificial intelligence. *Artif. Intell.* **2007**, *171*, 619–641. [[CrossRef](#)]
- Brannstrom, A.; Castagna, F.; Duchatelle, T.; Foulis, M.; Kampik, T.; Kuhlmann, I.; Malmqvist, L.; Morveli-Espinoza, M.; Mumford, J.; Pandzic, S.; et al. Online Handbook of Argumentation for AI: Volume 2. *arXiv* **2021**, arXiv:2106.10832.
- Modgil, S.; Prakken, H. The ASPIC⁺ Framew. Struct. Argum. A Tutorial. *Comput. Argum.* **2014**, *5*, 31–62. [[CrossRef](#)]
- Gordon, T.F.; Prakken, H.; Walton, D. The Carneades model of argument and burden of proof. *Artif. Intell.* **2007**, *171*, 875–896. [[CrossRef](#)]
- van Gijzel, B.; Prakken, H. Relating Carneades with abstract argumentation via the ASPIC⁺ framework for structured argumentation. *Comput. Argum.* **2012**, *3*, 21–47. [[CrossRef](#)]

19. Vassiliades, A.; Bassiliades, N.; Patkos, T. Argumentation and explainable artificial intelligence: A survey. *Knowl. Eng. Rev.* **2021**. [[CrossRef](#)]
20. Čyras, K.; Letsios, D.; Misener, R.; Toni, F. Argumentation for explainable scheduling. In Proceedings of the Conference AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33; pp. 2752–2759.
21. Nute, D. Defeasible Reasoning: A Philosophical Analysis in Prolog. In *Aspects of Artificial Intelligence*; Springer: Dordrecht, The Netherlands, 1988; pp. 251–288.
22. Governatori, G.; Rotolo, A.; Calardo, E. Possible World Semantics for Defeasible Deontic Logic. In *Deontic Logic in Computer Science*; Agotnes, T., Broersen, J., Elgesem, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 46–60.
23. Lam, H.P.; Governatori, G. The Making of SPINdle. In *Rule Interchange and Applications*; Governatori, G., Hall, J., Paschke, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 315–322.
24. Garcia, A.; Simari, G. Defeasible logic programming: An argumentative approach. *Theory Pract. Log. Program.* **2004**, *4*, 95–138. [[CrossRef](#)]
25. Dung, P.M. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artif. Intell.* **1995**, *77*, 321–358. [[CrossRef](#)]
26. Pisano, G.; Calegari, R.; Omicini, A.; Sartor, G. Arg-tuProlog: A tuProlog-based Argumentation Framework. In Proceedings of the Conference 35th Italian Conference on Computational Logic—CILC 2020, Rende, Italy, 13–15 October 2020; Volume 2710, pp. 51–66.
27. Calegari, R.; Contissa, G.; Pisano, G.; Sartor, G.; Sartor, G. Arg-tuProlog: A modular logic argumentation tool for PIL. In *Frontiers in Artificial Intelligence and Applications, Proceedings of the Legal Knowledge and Information Systems. JURIX 2020: The Thirty-Third Annual Conference, Brno, Czech Republic, 9–11 December 2020*; Villata, S., Harašta, J., Křemen, P., Eds.; IOS Press: Amsterdam, The Netherlands, 2020; Volume 334, pp. 265–268. [[CrossRef](#)]
28. Besnard, P.; García, A.J.; Hunter, A.; Modgil, S.; Prakken, H.; Simari, G.R.; Toni, F. Introduction to structured argumentation. *Argum. Comput.* **2014**, *5*, 1–4. [[CrossRef](#)]
29. Dung, P.M.; Sartor, G. The modular logic of private international law. *Artif. Intell. Law* **2011**, *19*, 233–261. [[CrossRef](#)]
30. Calegari, R.; Riveret, R.; Sartor, G. The Burden of Persuasion in Structured Argumentation. In Proceedings of the Conference Seventeenth International Conference on Artificial Intelligence and Law, São Paulo Brazil, 21–25 June 2020; Maranhão, J., Wyner, A.Z., Eds.; ACM: New York, NY, USA, 2021; pp. 180–184. [[CrossRef](#)]
31. Kampik, T.; Gabbay, D.; Sartor, G. The Burden of Persuasion in Abstract Argumentation. In Proceedings of the International Conference on Logic and Argumentation, Hangzhou, China, 20–22 October 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 224–243.
32. Riveret, R.; Rotolo, A.; Sartor, G. A Deontic Argumentation Framework Towards Doctrine Reification. *J. Appl. Logics-J. Logics5 Their Appl.* **2019**, *6*, 903–940.
33. Calegari, R.; Sartor, G. Burden of Persuasion in Argumentation. In Proceedings of the 36th International Conference on Logic Programming (Technical Communications), ICLP 2020, Rende, Italy, 18–24 September 2020.
34. Pisano, G.; Calegari, R.; Omicini, A. Towards cooperative argumentation for MAS: An actor-based approach. In *CEUR Workshop Proceedings, Proceedings of the WOA 2021–22nd Workshop “From Objects to Agents”, Bologna, Italy, 1–3 September 2021*; Calegari, R., Ciatto, G., Denti, E., Omicini, A., Sartor, G., Eds.; Sun SITE Central Europe, RWTH Aachen University: Aachen, Germany, 2021; Volume 2963, pp. 162–177.
35. Gordon, T.F.; Governatori, G.; Rotolo, A. Rules and norms: Requirements for rule interchange languages in the legal domain. In Proceedings of the International Workshop on Rules and Rule Markup Languages for the Semantic Web, Las Vegas, NV, USA, 5–7 November 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 282–296.