Legal Representation and Reasoning in Practice: A Critical Comparison

Conference Paper · October 2018

CITATIONS
0

READS
107

5 authors, including:

Sotirios Batsakis
University of Huddersfield
9 PUBLICATIONS **44** CITATIONS
SEE PROFILE

George Baryannis
University of Huddersfield
23 PUBLICATIONS **99** CITATIONS
SEE PROFILE

Guido Governatori
The Commonwealth Scientific and Industrial Research Organisation
368 PUBLICATIONS **6,072** CITATIONS
SEE PROFILE

Ilias Tachmazidis
University of Huddersfield
25 PUBLICATIONS **86** CITATIONS
SEE PROFILE

Legal Representation and Reasoning in Practice: A Critical Comparison

Sotiris BATSAKIS [c,a,1], George BARYANNIS [a], Guido GOVERNATORI [b], Ilias TACHMAZIDIS [a] and Grigoris ANTONIOU [a]

[a] *School of Computing and Engineering, University of Huddersfield, UK*
[b] *Data61, CSIRO, Brisbane, Australia*
[c] *Technical University of Crete, Greece*

Abstract. Representation and reasoning over legal rules is an important application domain and a number of related approaches have been developed. In this work, we investigate legal reasoning in practice based on three use cases of increasing complexity. We consider three representation and reasoning approaches: (a) Answer Set Programming, (b) Argumentation and (c) Defeasible Logic. Representation and reasoning approaches are evaluated with respect to semantics, expressiveness, efficiency, complexity and support.

Keywords. Legal knowledge representation, Legal reasoning, Normative reasoning

1. Introduction

Representation of legal rules and reasoning over them is a critical application area since laws and regulations are used in almost all human activities. Furthermore, corpora pertaining to laws and regulations are typically complex and enormous in size, thus experts are usually needed in order to apply legal reasoning in everyday life. Even for experts, this is a laborious, time-consuming, error-prone and costly process, which is why legal reasoning was one of the first application domains of artificial intelligence since its emergence, and more so after the proliferation of expert systems during the 1980s.

The logic-based structure of laws and regulations and the considerable benefits of automating the representation and reasoning processes has led to extensive research towards this direction. The importance and visibility of such research in recent years can be exemplified with the emergence of the field of Regulatory Technology (RegTech [1]). However, the complexity of the legal domain has made progress difficult. In civil law legal systems, which are the primary focus of this paper, several issues have been raised, such as: (a) the ambiguity of natural language used to express laws, which leads to different interpretations; (b) the existence of different conflicting rules that are applicable on the same case; and (c) the need to do reasoning in light of new information.

In this work, we identify and compare three distinct approaches that are capable of addressing some of these issues through features such as non-monotonic reasoning and

[1] Corresponding Author: Sotiris Batsakis, School of Computing and Engineering, University of Huddersfield, UK; E-mail: s.batsakis@hud.ac.uk.

conflict resolution. These are: Answer Set Programming, Argumentation and Defeasible Logic. Three use cases of increasing complexity are used for the comparison: a scenario on the presumption of innocence, licensing contractual clauses for the evaluation of a product and the reporting regulations on new drugs imposed by the US Food and Drug Administration (FDA). The main contributions of this work are the investigation of practical considerations of legal representation and reasoning through the aforementioned real-world use cases and a comparative evaluation of existing approaches and related tools used for these tasks. These, in turn, can assist in identifying strengths and weaknesses of various approaches, allowing for more informed choices on the appropriate solutions to different legal representation and reasoning problems.

This paper is organised as follows. A concise summary of efforts related to legal representation and reasoning is presented in Section 2. A description of the three use cases follows in Section 3. Representation and reasoning results using the three selected approaches are presented in Section 4 and they are critically evaluated in Section 5. Section 6 concludes and discusses directions for future research.

2. Background and Related Work

Early attempts at realising legal reasoning involved representing legislation in the form of Horn logic programs, subsequently extended with negation as failure, as in Sergot et al.'s seminal work on the British Nationality Act [2]. However, the treatment of double negation and counterfactual conditionals (e.g. "if it didn't rain") proved problematic. Moreover, exceptions in legislation are modeled explicitly by negative conditions in the rules [3], which is more suitable for self-contained and stable legislation but may require some level of rewriting whenever previously unknown exceptions (or chains of exceptions) are introduced.

Following the advent of the Semantic Web and the introduction of the OWL family of languages, several research efforts focused on examining whether description logics are a suitable candidate for representing and reasoning about legislation. A prime example is HARNESS [4], which shows that well-established sound and decidable description logic reasoners such as Pellet can be exploited for legal reasoning, if, however, a significant compromise in terms of expressiveness is made.

A common issue that arises when using classical or description logics in legal representation and reasoning is the fact that they are monotonic: logical consequences cannot be retracted, once entailed. This is in contrast to the nature of law, where legal consequences have to adapt in light of new evidence and conflicts between different regulations must be accounted for and resolved. Therefore, it is natural to employ non-monotonic logic for the purposes of legal reasoning. The Defeasible Logic framework [5] has been applied in a legal reasoning setting due to its simplicity and flexibility and the fact that several efficient implementations exist. In this framework, rules can either behave in the classical sense (*strict*), they can be defeated by contrary evidence (*defeasible*), or they can be used only to prevent conclusions (*defeaters*).

The notions of permission and obligation are inherent in normative reasoning but are not explicitly defined in traditional logic systems; deontic logic was introduced to serve this purpose. Permission and obligation are represented by modal operators and are connected to each other through axioms and inference rules. While there has been some

philosophical criticism on deontic logic due to its admission of several paradoxes (e.g. the gentle murderer), deontic modalities have been introduced to various logics to make them more suitable for normative reasoning. For instance, Governatori et al. [6] show how the aforementioned Defeasible Logic framework can be extended to model beliefs, intentions, obligations and permissions.

Legal reasoning, at its core, is a process of argumentation, with opposing sides attempting to justify their own interpretation, with appeals to precedent, principle, policy and purpose, as well as the construction of and attack on arguments [7]. AI and law research has addressed this with models that are based on Dung's [8] influential work, such as ASPIC+ [9], which offers means of producing argumentation frameworks tailored to different needs in terms of the structure of arguments, the nature of attacks and the use of preferences.

As argued by Brewka et al. [10], the high complexity of argumentation processes, including legal reasoning, makes them a prime application target for Answer Set Programming, a non-monotonic logic programming formalism, specifically aimed towards search problems of NP or higher complexity.

3. Use Cases

3.1. Use case 1: Presumption of Innocence

The first use case demonstrates the importance of the semantics of default inference in legal representation and reasoning. Presumption of innocence is the principle that one is considered innocent until proven guilty. Consider the following scenario in a legal case:

- Evidence A suggests that the defendant is not responsible.
- Evidence B suggests that the defendant is responsible.
- Sources for evidence A and B are equally reliable.
- According to the underlying legal system a defendant is presumed innocent (i.e. not guilty) unless responsibility has been proven (without any reasonable doubt).
- If the defendant is found to be innocent, they are entitled to compensation.

Given both evidences A and B, ambiguity exists as it is unclear whether the defendant should hold any responsibility. Thus, under this situation and with the underlying legal system, we should conclude that the defendant is not guilty and is entitled to compensation. However, if we allow the ambiguity with regard to responsibility to propagate, then the defendant's guiltiness should also be considered ambiguous; hence an undisputed conclusion cannot be drawn.

3.2. Use case 2: Smart Contracts in Blockchain Systems

The second use case, extracted from [11], is a typical example of legal reasoning related to contracts on blockchain systems and illustrates the intricacies inherent in the representation of the notions of permission and obligation. The following licensing contractual clauses are assumed for the evaluation of a product:

- Article 1. The Licensor grants the Licensee a licence to evaluate the Product.

- Article 2. The Licensee must not publish the results of the evaluation of the Product without the approval of the Licensor; the approval must be obtained before the publication. If the Licensee publishes results of the evaluation of the Product without approval from the Licensor, the Licensee has 24 hours to remove the material.
- Article 3. The Licensee must not publish comments on the evaluation of the Product, unless the Licensee is permitted to publish the results of the evaluation.
- Article 4. If the Licensee is commissioned to perform an independent evaluation of the Product, then the Licensee has the obligation to publish the evaluation results.
- Article 5. This licence will terminate automatically if Licensee breaches this Agreement.

Article 2 is of particular interest since it contains a reparation clause. Suppose that the licensee publishes the evaluation results without authorisation, then removes them within 24 hours. Then, according to Article 2, the licence to use the product still holds.

3.3. Use case 3: US FDA New Drugs Records and Reports Regulations

The third use case is much larger than the previous two; it includes legislation from the United States Electronic Code of Federal Regulations, specifically Title 21 - Food and Drugs, Part 310 - New Drugs, Subpart D - Records and Report. Specifically, the regulations define: (a) the reporting requirements for manufacturers, packers and distributors and (b) information on various life-threatening, serious or unexpected adverse drug experiences that have been reported in case safety reports. The legislation contains 8 paragraphs, with some of them containing as many as 12 requirements and can be found in [12]. For example, paragraph (c)(1) of the legislation requires that manufacturers, packers or distributors must report to FDA each adverse drug experience no later than 15 calendar days from initial receipt of complaint; however, this requirement is lifted if there is no reasonable possibility that the drug caused the adverse experience.

4. Representation and Reasoning

In this section, we present rule-based representations for the three use cases in the previous section, as well as reasoning results using these representations. The decision on candidate approaches was based on two fundamental criteria for practical legal reasoning: support for preferences over rules and availability of reasoning tools. Given these criteria, we selected the following: (1) Answer Set Programming (ASP), specifically Disjunctive Logic Programs with Inheritance [13]; (2) Argumentation, specifically structured argumentation within the ASPIC+ framework [9]; and (3) Defeasible Logic, specifically the extended version that supports deontic modalities [6].

4.1. Representation and Reasoning using ASP

ASP is an expressive form of Logic Programming based on the stable model semantics and supporting disjunction, among others. An extension introduced in [13] allows the expression of rule priorities by organising rules in inheritance networks and is imple-

mented by the DLV system [14]. Given two objects (rules or rule sets) o_1 and o_2, $o_2 : o_1$ denotes that o_2 is more specific than o_1 and overrides it, in case of conflict. Use case 1 is encoded as follows:

```
r1 {responsible:- evidenceA.}      r2 {-responsible:- evidenceB.}
r4 {-guilty:- 1=1.}                r3:r4 {quilty:-responsible.}
r6 {-compensation :- 1=1.}         r5:r6 {compensation:- -guilty.}
evidenceA.                         evidenceB.
```

Tautology 1=1 is required because objects in inheritance networks have to comprise rules and not facts. Running the DLV system using this encoding as an input yields no results (no stable models) because of the contradiction caused by rules r1 and r2; this is left unresolved and, as a result, no undisputed conclusions can be derived.

For use case 2, since deontic modalities are not supported natively by any ASP representation, we can only represent obligation explicitly within predicate names, while permission on a literal is encoded as a negated obligation on the literal's negation ($Oa \equiv \neg P \neg a$). The legislation of use case 2 is encoded as follows:

```
:- obl_publish, obl_not_publish.
art10 {obl_not_use :- 1=1.}
art11:art10 {-obl_not_use :- hasLicence.}
art21 {obl_not_publish :- 1=1.
       obl_remove :- publish, obl_not_publish.}
art22:art21 {-obl_not_publish :- hasLicense, hasApproval.}
art31 {obl_not_comment :- 1=1.}
art32:art31 {-obl_not_comment :- -obl_not_publish.}
art40:art21{obl_publish :- hasLicence,isCommissioned.
      -obl_not_publish :- hasLicence,isCommissioned.}
art51:art11 {obl_not_use :- violation.}
art52:art40,art22 {obl_not_publish :- violation.
                   -obl_publish :- violation.}
hasLicence.    publish.
```

The last two facts represent the case where the licensee has published results without having approval or being commissioned. Performing reasoning results in the stable model `{hasLicence, publish, -obl_not_use, obl_not_publish, obl_remove, obl_not_comment}`, which includes the permission to use, the obligations not to publish or comment and the obligation to remove published results. For use case 3, only the encoding for paragraph (c)(1) is presented due to space limitations. The full encoding contains 96 rules and can be found at https://github.com/gmparg/JURIX2018, along with all other encodings in this paper.

```
r5{obl_MPD_report_electronically_adverse_drug_exper_in_15_days
:- MPD_report_15_day_Postmarketing_Alert_reports.}
r6:r5{-obl_MPD_report_electronically_adverse_drug_exper_in_15_days
:-MPD_report_15_day_Postmarketing_Alert_reports_nocause.}
```

Note that if we are limited to the standard ASP format (ASP-Core-2), used by systems such as clingo [15] and DLV2 [16], then to compensate for the lack

of support for preferences over rules, extra negation-as-failure literals need to be included in rule bodies; for instance, to express in use case 2 that permission to use applies to a licensee unless they have violated the agreement, we use the rule `-obl_not_use :- hasLicence, not violation.` Reformulated encodings using negation as failure can also be found at https://github.com/gmparg/JURIX2018.

4.2. Representation and Reasoning using Argumentation

In the case of argumentation, using an abstract framework would offer only a coarse-grained level of representation which is not enough for fully representing a legal document. Structured argumentation, on the other hand, as is the case of argumentation according to ASPIC+, is capable of representing legislation since it supports encoding knowledge bases with axioms and premises. We used TOAST [17], an online ASPIC+ implementation. Use case 1 is encoded as follows (presented using TOAST's argumentation theory structure and syntax):

```
Premises: evidenceA; evidenceB;
Assumptions: default;
Rules:
[r1] evidenceA => responsible;      [r2] evidenceB => ~responsible;
[r3] responsible => guilty;         [r4] default=> ~guilty;
[r5] ~guilty => compensation;       [r6] default => ~compensation;
Rule Preferences: [r4] < [r3]; [r6] < [r5];
```

Note that rules in TOAST must have bodies, hence the inclusion of the default predicate. The computed extension contains only asserted facts *evidenceA* and *evidenceB* since arguments from r1 and r2 defeat each other, which is equivalent to the reasoning outcome in ASP. For use case 2, deontic modalities again need to be encoded explicitly, since deontic extensions like the one in [18] have not been implemented yet:

```
Premises: hasLicence; publish;
Assumptions: default;
Preferences: default < hasLicence; hasLicence < violation;
Rules:
[r1] default => obl_not_use;
[r2] hasLicence => ~obl_not_use;
[r3] default => obl_not_publish;
[r4] publish, obl_not_publish => obl_remove;
[r5] hasLicence, hasApproval => ~obl_not_publish;
[r6] default => obl_not_comment;
[r7] ~obl_not_publish => ~obl_not_comment;
[r8] hasLicence, isCommissioned => obl_publish;
[r9] hasLicence, isCommissioned => ~obl_not_publish;
[r10] violation => obl_not_use;
[r11] violation => obl_not_publish;
Rule Preferences:
[r1] < [r2]; [r3] < [r5]; [r6] < [r7]; [r3] < [r8]; [r3] < [r9];
[r2] < [r10]; [r5] < [r11]; [r8] < [r11]; [r9] < [r11];
```

```
Contrariness:
obl_publish-obl_not_publish;
```

The added preferences prevent implicit attacks caused by rules with contradictory heads; also, a contrariness relation is needed for every pair of arguments that are contradictory. Similarly to ASP, the computed extension includes the arguments for permission to use, obligation not to publish or comment and obligation to remove published results. The encoding of use case 3 contains the same amount of rules as the DLV encoding.

4.3. Representation and Reasoning using Defeasible Logic

For Defeasible Logic, we opted to use SPINdle [19], since it is the only reasoner that natively supports deontic extensions, which are necessary for use case 2. The encoding for use case 1 is as follows:

```
r1: evidenceA => responsible          r2: evidenceB => -responsible
r3: responsible => guilty             r4:  => -guilty
r5: -guilty => compensation           r6:  => -compensation
>> evidenceA                          >> evidenceB
r3 > r4                               r5 > r6
```

Note that Defeasible Logic allows expressing defeasible rules without bodies (presumptions). The reasoning output includes the following (with +/-D meaning definitely provable/not provable and +/-d meaning defeasibly provable/not provable: -D responsible(X), -D -responsible(X), -d responsible(X), -d -responsible(X), -D guilty(X), -D -guilty(X), +d -guilty(X), -d guilty(X), -D compensation(X), -D -compensation(X), +d compensation(X), -d -compensation(X). This means that, in contrast to DLV and TOAST, SPINdle infers the defeasible conclusion that the defendant is not guilty. This is due to the default ambiguity blocking behaviour in SPINdle; if we switch to ambiguity propagation, then the result will contain no derived conclusions, as in DLV and TOAST.

For use case 2, we use deontic modalities (Permission/Obligation) both on literals and rules, by adding [P] or [O] before the literal or at the end of a rule label. Syntax `label[O]: A=>B` is equivalent to `label: A=>[O]B`. The encoding is as follows:

```
Art10[O]: => -use(X)
Art11[P]: hasLicence(X) => use(X)
Art21[O]: => -publish(X)
Art21rep[O]: [O]-publish(X), publish(X) => remove(X)
Art22[P]: hasLicence(X), hasApproval(X) => publish(X)
Art31[O]: => -comment(X)
Art32[P]: [P]publish(X) => comment(X)
Art40[O]: hasLicence(X), isCommissioned(X) => publish(X)
Art40p[P]: hasLicence(X), isCommissioned(X) => publish(X)
Art51[O]: violation(X) => -use(X)
Art52[O]: violation(X) => -publish(X)
Art11 > Art10            Art22 > Art21            Art32 > Art31
Art40 > Art21            Art51 > Art11
Art52 > Art40            Art52 > Art40p           Art52 > Art22
>> hasLicence(X)         >> publish(X)
```

Table 1. Comparison of the presented approaches

	ASP	Argumentation	Defeasible Logic
Expressiveness	Negation As Failure, Disjunction	Negation As Failure	Presumptions, Deontic Modalities
Inconsistency Handling	Propositional	Direct/Indirect [9]	Paraconsistent [20]
Support (Reasoning)	DLV	TOAST	SPINdle
Complexity	Σ_2^P, Π_2^P [13]	P [21]	Linear [6]

Rule Art21rep implements the reparation clause of Article 2, while Art40p explicitly derives permission from obligation. The output includes the following defeasible derivations: +d [O]-comment(X), -d [P]comment(X), +d [O]-publish(X), -d [O]publish(X), -d [P]publish(X), +d [P]use(X), -d [O]-use(X), +d [O]remove(X). These correspond to the same obligations and permissions that are derived from the DLV and TOAST encodings. The encoding of use case 3 contains the same amount of rules as the previous encodings.

5. Critical Evaluation

In this section, we present a comparative evaluation of the three approaches based on the experience gained by representing the three use cases and reasoning with the represented knowledge. We begin with a summary of common characteristics. All three approaches operate under the semantics of closed-world assumption. In terms of expressiveness, they all support non-monotonic reasoning, classical negation and preferences between rules. All approaches are efficient is terms of reasoning time. Even in use case 3 all reasoners return results within two or three tenths of a second. Finally, in all three approaches, while a number of reasoning tools have been developed, there is currently no tool support for representation, to the best of our knowledge. This means that any legislation has to be manually encoded in the respective languages supported by the reasoners.

Table 1 summarises the differences among ASP, argumentation and Defeasible Logic, in what concerns legal knowledge representation and reasoning. In terms of expressiveness, Defeasible Deontic Logic is clearly the most efficient approach, supporting presumptions, permissions and obligations, which make representation an easier, faster and less error-prone process. In contrast, the other approaches have to compensate by expressing modalities explicitly within predicate names and by using default predicates or tautological rule bodies to model presumptions. In cases where legislation changes relatively often, adapting ASP and argumentation encodings would require comparatively more time and effort. It should be noted that while disjunction (supported by ASP) and negation as failure (supported by ASP and TOAST), do not feature in any of the use cases we examined, they may still be useful in encoding different types of legislation.

An important difference that manifests in use case 1 is the way the three approaches handle inconsistency. The pair of conflicting rules on responsibility lead to unsatisfiability in ASP, because DLV (and other reasoners) assume the standard propositional definition of consistency. TOAST behaves in the same way, with the computed grounded extension containing only the asserted facts, due to ASPIC+'s definition of direct consistency. SPINdle, on the other hand, infers the defeasible conclusion of not guilty, due to the paraconsistent nature of Defeasible Logic; this can be argued to be closer to common sense reasoning. Note that Defeasible Logic offers the flexibility to support both cases:

the aforementioned derivation is for the default ambiguity blocking semantics, while the alternative ambiguity propagation would not derive any undisputed conclusion.

Differences in inconsistency handling appear also in the case of contradicting facts in the knowledge base. ASP reasoners conclude that there is no stable model, while both argumentation and Defeasible Logic approaches allow the contradicting facts to be used in further inferences, ensuring however that no form of the principle of explosion is possible. Both of these ways to address inconsistent input can be useful, depending on whether the point is to detect inconsistencies or to derive results despite them. However, it should be noted that both TOAST and SPINdle lack some form of acknowledgment of the existence of inconsistency (e.g. a warning message), which would be fundamental in addressing cases where inconsistency is not desired, but is the result of mistaken input.

In terms of reasoning support, there is at least one stable reasoner available for each approach. SPINdle is the most recent, released in 2014, while TOAST and DLV were released in 2012. Though there are more recent ASP reasoners (e.g. clingo and DLV2), they do not support preferences over rules. There are also several other alternatives for argumentation (e.g. the Tweety libraries at http://tweetyproject.org/). In what concerns complexity, all algorithms implemented by the aforementioned reasoners are of polynomial time complexity, albeit at different points in the polynomial hierarchy: Defeasible Logic is at the lowest level (linear), followed by argumentation (P) and then ASP (NP). A more detailed comparison of ASPIC+ and Defeasible Logic can be found in [22].

6. Conclusions and Future Research Directions

We have presented a practical demonstration of how existing approaches and tools can be used for legal representation and reasoning. In terms of representation, all approaches are capable of encoding standard elements of legislation, though with different levels of efficiency depending on their expressive power. Reasoning results can vary, especially when conflicting rules without preferences or conflicting facts are included; in such cases particular attention is needed in order to employ the most appropriate approach (and provide the proper encoding) that corresponds to the intended behaviour.

Interesting research directions that can be derived from the presented critical comparison include: (a) improving existing reasoners in terms of reporting inconsistencies and support for deontic modalities; (b) exploring much larger use cases that may require some form of large-scale reasoning, as discussed in [23]; and (c) providing tool support for representation on top of the existing reasoners by exploiting research in legal natural language processing.

Acknowledgements

The authors wish to thank Robert A. Kowalski for his insightful comments on legal reasoning using logic programming. This work has been supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 690974 for the project MIREL: MIning and REasoning with Legal texts.

References

[1] P. Mann, RegTech: The Emergence of the Next Big Disruptor, *International Banker* (2017). https://internationalbanker.com/finance/regtech-emergence-next-big-disruptor.

[2] M.J. Sergot, F. Sadri, R.A. Kowalski, F. Kriwaczek, P. Hammond and H.T. Cory, The British Nationality Act as a Logic Program., *Commun. ACM* **29**(5) (1986), 370–386.

[3] R. Kowalski and A. Burton, WUENIC - A Case Study in Rule-Based Knowledge Representation and Reasoning., in: *JSAI-isAI Workshops*, M. Okumura, D. Bekki and K. Satoh, eds, Lecture Notes in Computer Science, Vol. 7258, Springer, 2011, pp. 112–125.

[4] S. Van de Ven, J. Breuker, R. Hoekstra and L. Wortel, Automated Legal Assessment in OWL 2., in: *JURIX*, E. Francesconi, G. Sartor and D. Tiscornia, eds, Frontiers in Artificial Intelligence and Applications, Vol. 189, IOS Press, 2008, pp. 170–175.

[5] G. Antoniou, D. Billington, G. Governatori and M.J. Maher, A Flexible Framework for Defeasible Logics., in: *AAAI/IAAI*, H.A. Kautz and B.W. Porter, eds, AAAI Press / The MIT Press, 2000, pp. 405–410.

[6] G. Governatori, F. Olivieri, A. Rotolo and S. Scannapieco, Computing Strong and Weak Permissions in Defeasible Logic., *J. Philosophical Logic* **42**(6) (2013), 799–829.

[7] H. Prakken and G. Sartor, Law and logic: A review from an argumentation perspective., *Artif. Intell.* **227** (2015), 214–245.

[8] P.M. Dung, On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games, *Artificial Intelligence* **77**(2) (1995), 321–357.

[9] S. Modgil and H. Prakken, The ASPIC+ framework for structured argumentation: a tutorial, *Argument & Computation* **5**(1) (2014), 31–62. doi:10.1080/19462166.2013.869766.

[10] G. Brewka, M. Diller, G. Heissenberger, T. Linsbichler and S. Woltran, Solving Advanced Argumentation Problems with Answer-Set Programming, in: *AAAI*, 2017.

[11] G. Governatori, F. Idelberger, Z. Milosevic, R. Riveret, G. Sartor and X. Xu, On legal contracts, imperative and declarative smart contracts, and blockchain systems, *Artificial Intelligence and Law* (2018).

[12] US Government, Records and reports concerning adverse drug experiences on marketed prescription drugs for human use without approved new drug application, 2014. http://bit.ly/eCFR310_305.

[13] F. Buccafurri, W. Faber and N. Leone, Disjunctive Logic Programs with Inheritance, *Theory Pract. Log. Program.* **2**(3) (2002), 293–321.

[14] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri and F. Scarcello, The DLV System for Knowledge Representation and Reasoning, *ACM Trans. Comput. Logic* **7**(3) (2006), 499–562.

[15] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub and P. Wanko, Theory Solving Made Easy with Clingo 5, in: *Technical Communications of the 32nd International Conference on Logic Programming (ICLP'16)*, M. Carro and A. King, eds, OASIcs, Vol. 52, Schloss Dagstuhl, 2016, pp. 1–15.

[16] M. Alviano, F. Calimeri, C. Dodaro, D. Fuscà, N. Leone, S. Perri, F. Ricca, P. Veltri and J. Zangari, The ASP System DLV2, in: *Logic Programming and Nonmonotonic Reasoning*, M. Balduccini and T. Janhunen, eds, Springer International Publishing, 2017, pp. 215–221.

[17] M. Snaith and C. Reed, TOAST: Online ASPIC+ implementation, in: *Proc. of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, B. Verheij, S. Szeider and S. Woltran, eds, Frontiers in Artificial Intelligence and Applications, Vol. 245, IOS Press, 2012.

[18] L.W.N. van der Torre and S. Villata, An ASPIC-based legal argumentation framework for deontic reasoning., in: *COMMA*, S. Parsons, N. Oren, C. Reed and F. Cerutti, eds, Frontiers in Artificial Intelligence and Applications, Vol. 266, IOS Press, 2014, pp. 421–432. ISBN 978-1-61499-436-7.

[19] H.-P. Lam and G. Governatori, The Making of SPINdle., in: *RuleML*, G. Governatori, J. Hall and A. Paschke, eds, Lecture Notes in Computer Science, Vol. 5858, Springer, 2009, pp. 315–322.

[20] M.J. Maher, A Model-Theoretic Semantics for Defeasible Logic., in: *Paraconsistent Computational Logic*, H. Decker, J. Villadsen and T. Waragai, eds, Datalogiske Skrifter, Vol. 95, 2002, pp. 67–80.

[21] P.E. Dunne and M. Wooldridge, Complexity of Abstract Argumentation, in: *Argumentation in Artificial Intelligence*, I. Rahwan and G.R. Simari, eds, Springer-Verlag, 2009, pp. 85–104.

[22] H.-P. Lam, G. Governatori and R. Riveret, On ASPIC+ and Defeasible Logic., in: *COMMA*, P. Baroni, T.F. Gordon, T. Scheffler and M. Stede, eds, Frontiers in Artificial Intelligence and Applications, Vol. 287, IOS Press, 2016, pp. 359–370. ISBN 978-1-61499-686-6.

[23] G. Antoniou, G. Baryannis, S. Batsakis, G. Governatori, L. Robaldo, G. Siragusa and I. Tachmazidis, Legal Reasoning and Big Data: Opportunities and Challenges, in: *MIREL 2018 Workshop*, 2018. https://doi.org/10.29007/tkmv.