XAI-LAW Towards a logic programming tool for taking and explaining legal decisions*

Agostino Dovier[1,2], Talissa Dreossi[1,2] and Andrea Formisano[1,2]

[1]Dip. di Scienze Matematiche, Informatiche e Fisiche, Università degli Studi di Udine, 33100 Udine, Italy
[2]GNCS-INdAM, Gruppo Nazionale per il Calcolo Scientifico.

Abstract

In this paper we present an overview of a research project aiming at producing a semi-automated tool for legal reasoning in the Italian criminal system during the criminal trial stage.

Parts of the Italian Criminal Law are modeled in ASP; the model obtained is tested on a set of previous statements on the crimes, and, if needed, refined. Decisions on a new case can be suggested by the system and explained using a tool that exploits "supportedness" of stable models. In the same way, the decision of a judge can be input in the system and automatically explained. Using a system of inductive logic programming for ASP, the tool can evolve by analyzing new statements and performing model revision, by learning exceptions, and by applying rule generalization. To study feasibility of the approach we analyzed the crimes of theft, robbery, and personal injuries. Further crimes will be considered in the future development of the project.

Keywords

Automated Reasoning, Non Monotonic Reasoning, Legal reasoning, Logical Learning

1. Introduction

The desire of delegating legal decisions to an automated formal system can be traced back (at least) to Leibniz's dream. In spite of Gödel incompleteness results, holding in a more general context, several efforts have been posed in this direction since the birth of artificial intelligence, earlier expert systems, and logic programming. Allen [1] in the Fifties moved the first steps towards the interpretation of legal documents in symbolic logic. In this seminal work he also pointed out which are the main logical connectives needed for this purpose and how using them. Thirty years later Allen, with Saxon [2] explained in details two features of legal rules that require attention in logical encoding. One of them is related to the inherent ambiguity of natural language —often exploited by smart lawyers— that allows one multiple semantic interpretation of the same sentence. To address this aspect, various efforts have been made by researchers that led to the proposal of Logical English (LE) for law (and education) [3]. The second issue concerns law rule applicability: sometimes rules are applied even if their preconditions have not been completely proved, or rules admit commonsense exceptions, and so on. This was really an issue for expert systems in the Eighties but it is, instead, a feature in reasoners based on stable model semantics.

Kowalski and Sergot [4] in the same years classified (legal) rules into definitional, normative, and case law and succeeded in representing in logic programming *a significant portion of several British laws in Horn clause form and various of its extensions* (and, in particular, a significant portion of the 1981 British Nationality Act). This approach proved that logic programming can be used for modeling legal rules and reasoning. Even if the reasoning modules at that time were not suited for a not stratified use of default negation in modeling.

Golshani [5] stated that: *Unlike ordinary expert systems, an automated legal reasoning system does not aim to provide an answer. Its objective should be to provide, for any given case, a well-constructed*

CILC 2024: 39th Italian Conference on Computational Logic, June 26-28, 2024, Rome, Italy

*Research partially supported by Interdepartment Project on AI (Strategic Plan of UniUD–2022-25), by MaPSART-FAIR project, and by GNCS 2024 project LCXAI: Logica Computazionale per eXplainable Artificial Intelligence.

agostino.dovier@uniud.it (A. Dovier); talissa.dreossi@uniud.it (T. Dreossi); andrea.formisano@uniud.it (A. Formisano)
0000-0003-2052-8593 (A. Dovier); 0009-0007-1746-6500 (T. Dreossi); 000-0002-6755-9314 (A. Formisano)

argument (or preferably several of them) rather than a definitive answer. The judge needs to be the one that takes the decision. But She/He might accept hypotheses to be analyzed.

Modern systems based on ASP can generate several scenarios (i.e., stable models). Moreover, each of them, being *supported* models, can explain the reasons of the presence of atoms in the model.

In some countries (e.g., UK) law is mostly based upon previous cases. Precedents have no legal authority, but are considered for the sake of consistency and fairness. This view seems to encourage machine learning approaches. However, these approaches cannot learn the logic behind the reasons for the choices. Moreover, in other countries (e.g., Italy) cases are used only in presence of ambiguity of the interpretation of the law that, instead, is formally written in codes. Cases can be used for learning exceptions.

This leads towards a logic approach that exploits the modern computation capability of Conflict Driven Clause Learning (CDCL) bottom-up ASP solvers. The learning capability, which is surely important, can be retrieved by making use of modern tools based on inductive logic programming and tailored for ASP, such as ILASP [6]. Handling exceptions is one of the main features of systems developed for non-monotonic reasoning.

The recent contribution of [7] proves that the use of Logical English as an intermediate language, helps in encoding laws into ASP programs. The authors of [7] use the top-down ASP solvers s(CASP) since it is equipped with an explanation tool (acting top-down it just computes the paths that lead to the solution, instead of a complete model of the ASP encoding). Top-down approaches to ASP solving have two main drawbacks. The first is scalability since they cannot exploit the CDCL technique for pruning the search tree. The second is more subtle, namely they can return a consequence even in case of inconsistent programs (namely, those not admitting any stable model). To detect inconsistency, other portions of the program should be analyzed.

In our project we will use ASP encoding of immutable legal rules, bottom-up CDCL ASP solvers, and tools for explainability of the stable models. Dynamic refinement of the model by cases, either by hand or automatically with ILASP, is used for a continuous improvement of our proposal. The overall system will be capable of computing scenarios of plausible decisions, each of them endowed with a detailed explanation, that can be analyzed by a judge for taking her/his decision, or it can be used to find a logical explanation of a decision autonomously taken by a judge. Although, in principle, the system could be used for autonomous decision-making, the authors would suggest its use as a decision support system.

2. System Description

We assume the reader is aware of the basic terminology of logic programming, in particular in the context of non monotonic reasoning with stable model semantics. We briefly describe the main tasks of the project we are presenting:

Criminal law encoding. This stage is carried on, currently, by humans. The proponents and their collaborators are encoding various parts of the Italian Criminal Code (ICC)[1] in ASP. If a rule could be open to exceptions (even if not explicitly written in the law), this should be reflected in the definition.

As an instance, the rule "*if you subtract an object owned by another person you are guilty of theft*" can be encoded in ASP as follows:

```
guilty_theft(R) :-
       subtract(R,C), own(V,C), subtracted_obj(C),
       not extraordinary(R,C,V).
```

The last literal in the body is added to permit exceptions. For instance, it can be the case that R was claimed to be owned by V, but the reality is that it was just stolen by V to another person, and R is a policewoman. Unless it turns out later that R was Eva Kant dressed as a policewoman.

[1] https://www.gazzettaufficiale.it/sommario/codici/codicePenale

Exception predicates can be defined since the beginning and extended in subsequent stages when new information is known by previous court rulings.

Vagueness handling. In this stage -actually closely related to the previous one- vague concepts can be dealt with by exploiting non-deterministic choices. For instance, as presented in [8], the difference between robbery and snatch theft (e.g., in stealing a handbag) lays in the level of adhesion of the object to the body of the victim. However, in the ICC there is no quantitative information (e.g., amount of distance) for determining whether there is adhesion or not. One could express this using an ASP rule of the form

```
{adherence(V,C)} :-
       unknown_adherence(V,C), subtracted_obj(C), victim(V).
```

Or, using a finer granularity level:

```
1{adherence(V,C,L): level(L)}1 :-
       unknown_adherence(V,C), subtracted_obj(C), victim(V).
```

With these non-deterministic rules, more alternative hypotheses are considered by the ASP solver. During the procedural trial it can be used also for explaining the motivations made (independently) by a judge; new information can be used to remove some of them by adding denials. For instance, if there is a picture showing that there is enough room between the handbag and waist of the victim, one can add the ASP constraint:

```
:- adherence(V,C,L), level(L), L < 2, subtracted_obj(C), victim(V).
```

Model verification and static refining. As outcome of the previous two stages, we obtain an ASP program P_{law} modeling some juridical knowledge/principles. We can perform a refining step aimed at improving the quality of the model. A set of known criminal cases, taken from two main databases (Foroplus and DeJure[2]), are then analyzed and encoded by hand in a set of facts $C_1, \ldots, C_k$. For $i = 1, \ldots, k$, the ASP program $P_{law} \cup C_i$ is processed to compute its stable models. If it is does not admit any stable model, the issue is analyzed and the P_{law} should be refined (rules were probably too strong).

If it admits a stable model, then let $\neg C_i$ be the set of denials

```
:- not atom.
```

for every true `atom` in C_i. Then, one can try to find the stable models of P_{law} with $\neg C_i$ to check if the desired model is obtained. If not, P_{law} should be refined (some rules were probably too weak).

When everything works, it is interesting to verify whether, assuming a subset of C_i facts hold, models different from C_i are generated. Exceptions extending P_{law} might emerge in this stage, as well.

Model explanation. This is the first stage that can be implemented automatically. Given the program P_{law} and a sentence C, after run the ASP solver on P_{law} and $\neg C$, one can analyze the relevant set of atoms in the stable model; Then, the rules explaining why these atoms are in the stable model are emphasized (e.g., with a specialization of the approach presented by Alviano et al [9]).

Thus, the tool can be used also for explaining the motivations made (independently) by a judge. But in a second moment it can be used by a judge (or by the lawyers involved in the procedural venue) to generate some possible scenarios, each of them endowed with an explanation ("motivazioni della sentenza" in Italian).

For instance, consider the following (real) judgment (Cassazione penale sez. II, 21/02/2019, n.16899): "Case in which the Court deemed correct the qualification as robbery of the actions committed by the

[2] https://www.foroplus.it/home, https://dejure.it/#/home

defendant who, approaching the elderly victims from behind, grabbed their heads and immobilized them with a compression maneuver, ensuring the necessary immobility to remove earrings from the victims' earlobes". This situation can be encoded as outlined below (names of reo and victim are fictitious):

```
own("Veronica", "earrings").
subtract("Giulio", "earrings").
snatch("Giulio", "earrings").
take_possession("Giulio", "earrings").
```

Moreover, it can be assumed that the adherence of the object is tight since earrings are locked to the earlobes. This is encoded as: `adherence("Veronica", "earrings", 4)`. Running the solver will end in: `robbery("Giulio", "Veronica")`. The explanation is shown in Figure 1. The rounded pink boxes contain the facts, what actually happened, while the green ones contain the deduced facts. The light blue boxes display the rules used to make these deductions. Arrows have different colors just for readability.

Furthermore, as part of our ongoing work, we are modeling two additional articles of the Italian legal code: Article 581 and Article 582, which deal with beatings and personal injuries. The following example illustrates the progress we have made this far. Given:

```
harmful_intention("Giulio").
damage("Giulio", "Veronica").
derive("Veronica", "Ferita").
```

where `derive` means that the victim derives an illness from the damage caused by the attacker, the solver is able to infer: `personal_injuries("Giulio","Veronica","Ferita")`.

Dynamic Model refining. A second stage that can be automated or at least semi-automated is a continuous, successive refining of the tool when new sentences on the crimes are known and added to P_{law} In this case, the issue is the one of translating the -rather technical- legal text explaining a sentence, in ASP facts and (sometimes) rules. The work done in the previous stages should have shown which are the main predicates that can be affected by program revision. This information is an input for ILASP, the Inductive Logic Programming System for ASP [6] that can be run to create an extended P_{law} program where new rules are added and, sometimes, other rules are removed by generalization. Results on experiments of ILASP learning capability can be found in the last edition of the CILC conference [10]. A human check can be required here if some flag is raised (e.g., when partial or total model inconsistency with previously tested sentences is detected).

Dealing with second and third justice levels. The Italian criminal law systems is based on three levels (*1. primo grado, 2. appello, 3. cassazione*). In principle, higher degree might change the results of the previous levels.

However, the three trials are not equal. We cannot be exhaustive here in classifying the differences. Just as an example, consider that the second level might change the decisions of the first level if there is evidence of some fact not available before (that therefore change completely the deduced stable models and the explanation) or it might underline the inadequacy of the reasoning made by the first judge. The last case could be interesting from out point of view since it would be a statement against the fact that the stable model is supported. This kind of evidence can be made by our system against first level sentences.

Reasons for invalidating sentences at the third level are more related to errors of using the judicial process rules. This is not directly related to the P_{law} program. For instance the document that proves the guilty for some privacy reasons could not be used as a legal proof.

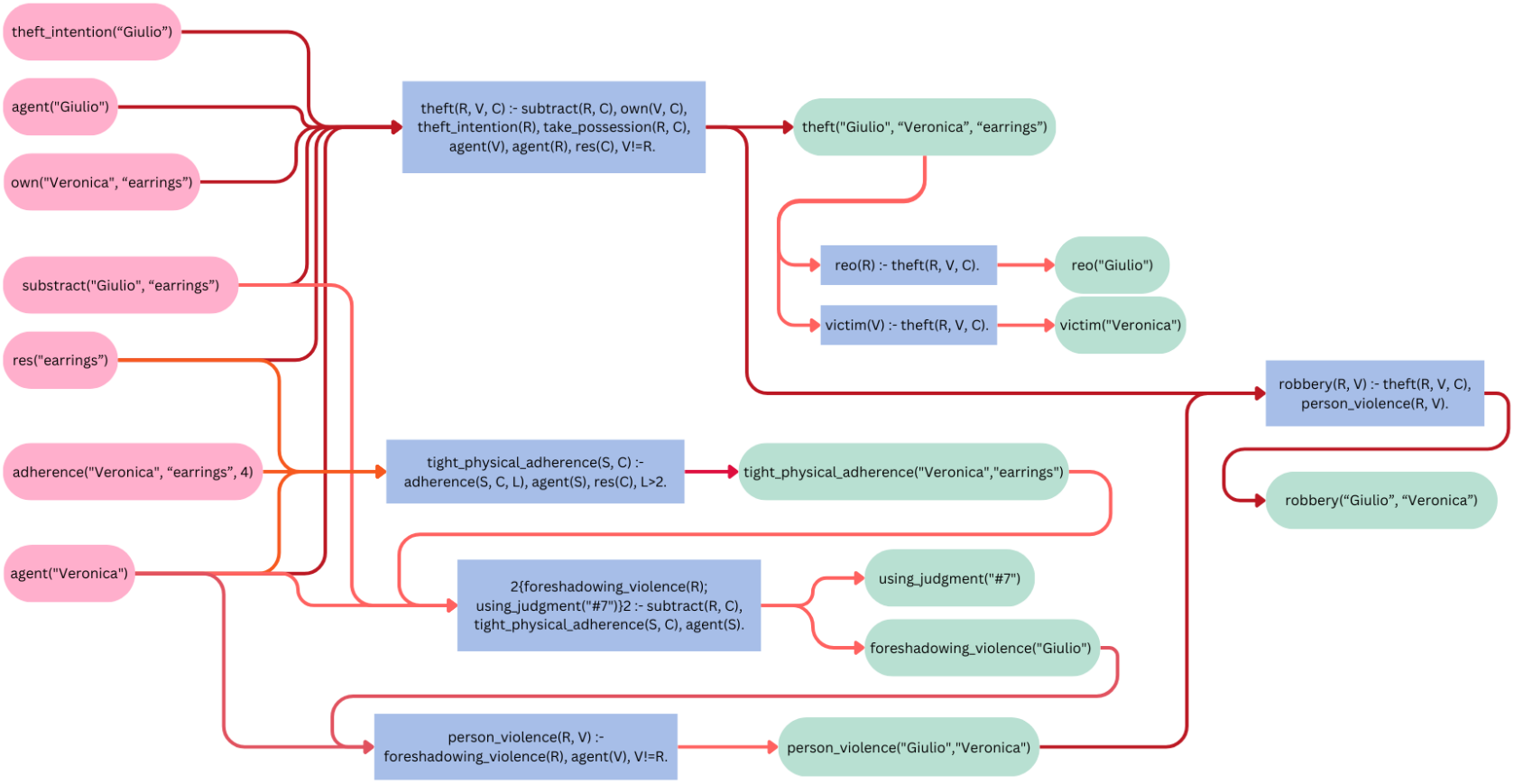

Figure 1: Explanation of judgment

3. Conclusions

In this short contribution we described the main tasks composing a research activity we are currently developing. The main themes of the project focus on the automation of analysis, learning, and reasoning about juridical knowledge (juridical cases, laws, judgments, trials processes, etc), by exploiting inductive logic programming techniques and tools. The main advantage in adopting logic programming, compared to, for example, using ML-based techniques, is the immediate explainability properties of the system. As an example, we mention a recent approach exploiting ASP to model and mechanize steps of the investigative process described in, e.g., [11]. Actually, approaches such as the one of [11] can be combined with the framework that is expected as product of our project. Focusing on the criminal trial stage, as mentioned, first attempts have been described in [8, 12]. The choice of ILASP as inference engine opens further lines of research. For instance, we intend to apply to the ILASP solver the ASP-technology developed in parallelizing ASP-solvers (cf., [13, 14, 15, 16]) in order to improve ILASP efficiency and scalability. As far as the explainability issues we will relate our results to those produced by argumentation-based approaches (e.g., [17]).

Acknowledgments

The authors would like to thank all other colleagues and students involved in the project. In particular:

- Alessandra Russo and Mark Law, for introducing us to ILASP and its variants and for their kindness and guidance when TD visited Imperial College
- Luca Baron, Manuele Dozzi, and Federico Costantini, for sharing their insights of the world of Italian criminal law reasoning allowing the starting of a real interdisciplinary project
- Benedetta Strizzolo and Lorenzo Cian, for their hard and clever work (hoping that it is just the beginning of a long and fruitful collaboration).

References

[1] L. E. Allen, Symbolic logic: A razor-edged tool for drafting and interpreting legal documents, The Yale Law Journal 66 (1957) 933–879.

[2] L. E. Allen, C. S. Saxon, Some problems in designing expert systems to aid legal reasoning, in: Proceedings of the First International Conference on Artificial Intelligence and Law, ICAIL '87, Boston, MA, USA, May 27-29, 1987, ACM, 1987, pp. 94–103. URL: https://doi.org/10.1145/41735.41747. doi:10.1145/41735.41747.

[3] R. A. Kowalski, J. A. Dávila, G. Sartor, M. Calejo, Logical english for law and education, in: D. S. Warren, V. Dahl, T. Eiter, M. V. Hermenegildo, R. A. Kowalski, F. Rossi (Eds.), Prolog: The Next 50 Years, volume 13900 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 287–299. URL: https://doi.org/10.1007/978-3-031-35254-6_24. doi:10.1007/978-3-031-35254-6_24.

[4] R. A. Kowalski, M. J. Sergot, Computer representation of the law, in: A. K. Joshi (Ed.), Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985, Morgan Kaufmann, 1985, pp. 1269–1270. URL: http://ijcai.org/Proceedings/85-2/Papers/114.pdf.

[5] F. Golshani, Automated construction of legal arguments, Int. J. Intell. Syst. 6 (1991) 673–685. URL: https://doi.org/10.1002/int.4550060605. doi:10.1002/INT.4550060605.

[6] M. Law, Inductive learning of answer set programs, Ph.D. thesis, Imperial College London, 2018.

[7] G. Sartor, J. A. Dávila, M. Billi, G. Contissa, G. Pisano, R. A. Kowalski, Integration of logical English and s(CASP), in: J. Arias, R. Calegari, L. Dickens, W. Faber, J. Fandinno, G. Gupta, M. Hecher, D. Inclezan, E. LeBlanc, M. Morak, E. Salazar, J. Zangari (Eds.), Proceedings of the International Conference on Logic Programming 2022 Workshops co-located with the 38th International Conference on Logic Programming (ICLP) 2022, Haifa, Israel, July 31st - August 1st, 2022, volume 3193 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3193/paper5GDE.pdf.

[8] M. Dozzi, T. Dreossi, F. Costantini, A. Dovier, A. Formisano, Semi-automatic knowledge representation and reasoning on vagueness crime concepts, in: ALP2023, JURIX 2023 workshop on AI, law and philosophy, Maastricht, Netherlands, 2023.

[9] M. Alviano, L. L. T. Trieu, T. C. Son, M. Balduccini, Advancements in xASP, an XAI system for answer set programming, in: A. Dovier, A. Formisano (Eds.), Proceedings of the 38th Italian Conference on Computational Logic, Udine, Italy, June 21-23, 2023, volume 3428 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3428/paper2.pdf.

[10] T. Dreossi, Exploring ILASP through logic puzzles modelling, in: A. Dovier, A. Formisano (Eds.), Proc. of CILC-23, volume 3428 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.

[11] S. Costantini, G. D. Gasperis, R. Olivieri, Digital forensics and investigations meet artificial intelligence, Ann. Math. Artif. Intell. 86 (2019) 193–229. URL: https://doi.org/10.1007/s10472-019-09632-y. doi:10.1007/S10472-019-09632-Y.

[12] M. Dozzi, T. Dreossi, L. Baron, F. Costantini, A. Dovier, A. Formisano, Semi-automatic knowledge representation and reasoning on vague crime concepts, in: Workshop DigForASP, colocated in 15th European Symposium on Computational Intelligence and Mathematics, Krakow, Poland, 2024.

[13] A. Dovier, A. Formisano, E. Pontelli, F. Vella, A GPU implementation of the ASP computation, in: M. Gavanelli, J. H. Reppy (Eds.), Proc. of PADL 2016, volume 9585 of *LNCS*, Springer, 2016, pp. 30–47. DOI:10.1007/978-3-319-28228-2_3.

[14] A. Dovier, A. Formisano, E. Pontelli, Parallel answer set programming, in: Y. Hamadi, L. Sais (Eds.), Handbook of Parallel Constraint Reasoning, Springer, 2018, pp. 237–282. DOI:10.1007/978-3-319-63516-3_7.

[15] A. Dovier, A. Formisano, F. Vella, GPU-based parallelism for ASP-solving, in: P. Hofstedt, S. Abreu, U. John, H. Kuchen, D. Seipel (Eds.), Declarative Programming and Knowledge Management - DECLARE 2019, Revised Selected Papers, volume 12057 of *LNCS*, Springer, 2019, pp. 3–23. DOI:10.1007/978-3-030-46714-2_1.

[16] A. Dovier, A. Formisano, G. Gupta, M. V. Hermenegildo, E. Pontelli, R. Rocha, Parallel logic programming: A sequel, Theory Pract. Log. Program. 22 (2022) 905–973. URL: https://doi.org/10.1017/S1471068422000059. doi:10.1017/S1471068422000059.

[17] H. Prakken, G. Sartor, Law and logic: A review from an argumentation perspective, Artif. Intell. 227 (2015) 214–245. URL: https://doi.org/10.1016/j.artint.2015.06.005. doi:10.1016/J.ARTINT.2015.06.005.