Modeling Administrative Discretion Using Goal-Directed Answer Set Programming [⋆]

Joaquín Arias[1 [0000−0003−4148−311X]], Mar Moreno-Rebato[1 [0000−0002−4177−9239]], Jose A. Rodriguez-García[1 [0000−0002−6362−9880]], and Sascha Ossowski[1 [0000−0003−2483−9508]]

CETINIA, Universidad Rey Juan Carlos, Madrid, Spain

Abstract. Automated legal reasoning and its application in smart contract is getting interest. In this context, ethical and legal concerns make it necessary for automated reasoners to *justify* in human-understandable terms the advice given. Logic Programming, specially Answer Set Programming, has a rich semantics and has been used to very concisely express complex knowledge. However, modelling vague concepts such as *ambiguity* and *discretion* cannot be expressed in top-down execution models based on Prolog, and in bottom-up execution models based on ASP the justifications are incomplete and/or not scalable. We propose to use s(CASP), a top-down execution model for predicate ASP, to model ambiguity and discretion following a set of patterns. We have implemented a framework, called s(LAW), to model, reason, and justify the applicable legislation and validate it by translating (and benchmarking) the criteria for the admission of students in public centers established by the "Comunidad de Madrid".

Keywords: Answer Set Programming · Goal-Directed · Ambiguity · Administrative Discretion.

1 Introduction

The formal representation of a legal text to automatize reasoning about them is well known in literature. For deterministic rules there are several proposals, often based on logic-based programming languages [12,14].

This topic is recently gaining much attention thanks to the interest in the so-called smart contracts, and to autonomous decisions by public administrations [10,3,15]. A smart contract is a program that represents the legal terms of a contract and is deployed on a block-chain platform to automatically execute, control and document the events described in the contract.

However, none of the existing proposals are able to represent the ambiguity and/or administrative discretion present in contracts and/or applicable legislation, e.g., *force majeure*. Force majeure is a law term that must be understood as referring to abnormal and unforeseeable circumstances which were outside the control of the party by whom it is pleaded and the consequences of which could not have been avoided in spite of the exercise of all due care (see

[⋆] This work has been partially supported by the Spanish Ministry of Science, Innovation and Universities, co-funded by EU FEDER Funds, through project grant InEDGEMobility RTI2018-095390-B-C33 (MCIU/AEI/FEDER, UE).

judgment Court of Justice of European Union, case Tomas Vilkas, C-640/15, 25 January 2017). In the procedure for awarding school places in centers supported with public funds in the "Comunidad de Madrid" (CM), in Spain, the proximity of a school to a family's home or work address plays an important role. This proximity is determined based on existing educational districts, except in cases of force majeure, but these cases are not defined a priory.

In this work we present a framework, called s(LAW), that allows for modeling legal rules involving ambiguity, and supports reasoning and inferring conclusions based on them. Additionally, thanks to the goal-directed execution of s(CASP), the underlying system used to implement our proposal, s(LAW) provides justification of the resulting conclusions (in natural language).

To evaluate the expressiveness of our proposal we have translated the procedure for awarding school places for the "Educación Secundaria Obligatoria" (ESO) of centers supported with public funds in the CM. The Spanish Organic Law on Education[1] regulates, in article 84, the criteria for the admission of students in public centers and private subsidized centers and, in its second paragraph of this article 84, indicates adjudication criteria. However, since Spain is a politically decentralized country, it is the autonomous communities (and, therefore, their educational administrations) that have powers to develop these aspects of basic state legislation. The CM, in use of its powers in educational matters, establishes the framework and general procedure for the admission of students to educational centers supported with public funds for the ESO.[2] The case presented in this paper is, therefore, a real case, based on the regulations currently in force in the CM.

2 Goal-Directed Answer Set Programming

Our proposal relies on Answer Set Programming (ASP) [7] for coding contracts and legal rules. More specifically, we use s(CASP)[1], a goal-directed implementation of ASP that features predicates, constraints among non-ground variables, and uninterpreted functions.

The top-down query-driven execution strategy of s(CASP) has three major advantages w.r.t. traditional ASP system: (a) it does not require to ground the programs; (b) its execution starts with a query and the evaluation only explores the parts of the knowledge base relevant to the query; and (c) s(CASP) returns partial stable models (the relevant subsets of the ASP stable models needed to support the query) and their corresponding justification (proof tree). Thus,

[1] Organic Law 2/2006, May 3, last modified by Organic Law 3/2020, December 29

[2] Decree 29/2013, of April 11, modified by Decree 11/2019, of March 5, of the Governing Council, on freedom of choice respecting school centers; Order 1240/2013, of April 17, of the Department of Education, Youth and Sports of Community of Madrid, modified by Order 1534/2019, of May 17, of the Department of Education and Research Community of Madrid; Resolution of July 31, 2013, of the General Directorate for the Improvement of the Quality of Education (regarding bilingual education); and Joint Resolution of the Deputy Department of Educational Policy and Educational Organization, of February 18, 2021 (https://bit.ly/3dAX22d).

our proposal automates commonsense reasoning and is scalable whereas ground based ASP systems do not (Section 5).

Additionally, s(CASP) provides a mechanism to present justifications in natural language using a generic translation, and the possibility of customizing them with directives that provide explanation patterns in natural language. Both plain text and user-friendly, expandable HTML can be generated. These patterns can be used with the program text itself, thereby making it easier for experts without a programming background to understand both the program and the results, i.e., partial model and justification, of its execution.

3 Administrative and judicial discretion reasoner

This work makes two main contributions: (i) a set of patterns to translate legal rules into ASP, and natural language patterns to generate readable justifications; (ii) a framework to model, reason, and justify conclusions based on the evidence provided by the user and the applicable legislation, representing ambiguity, discretion and/or incomplete information (key concepts in legal cases).

3.1 Patterns to translate law into ASP

The translation of legal rules into logic predicates has been considered a straightforward task for many years. However, the translation of ambiguity and/or discretion concepts required the help of an expert in law and/or in the field of application, in order to specify only one interpretation and/or decision.

Let us use the encoding of the procedure for the adjudication of schools places in the CM (Fig. 1) to explain the following patterns:

Requirement For Applying These are the most common constructions in legal articles. There are two patterns:

– Disjunction of requirements, e.g., “s/he obtains a school place if one of the following common requirements are met”. Which is translated by separating each requirement in different clauses, see Fig. 1 lines 9, 12, and 19:
– Conjunction of requirements, e.g., “In addition, some of the specific requirements must be met”. Which is translated to a single clause where the comma `','` means *and*, see Fig. 1 lines 5-7:

Exceptions For Applying As we mentioned before, a legal article is a default rule subject to possible exceptions. In s(CASP) the exceptions can be encoded using negation as failure. For example, Fig. 1 lines 2-4 shows the translation of “It will be possible to obtain a school place if the requirement is met and there is no exception” and then, the compiler of s(CASP) would generate its dual, i.e., `not exception`, by collecting and checking that no exceptions hold:

```
not exception :- not exception_1, ..., not exception_n.
```

where `not exception_i` is a new predicate name that identified the dual of the ith exception. For the sake of brevity let us omit the explanation of how the compiler generates the dual for each exception (see [9,1] for details). Fig. 1 lines 46-57 shows the translation of the unique exception defined in our running

```
%% Obtain a school place if...
obtain_place :-
    met_requirement,
    not exception.
met_requirement :-
    met_common_requirement,
    met_specific_requirement.
%% Common requirements:
met_common_requirement :-
    large_family.

met_common_requirement :-
    recipient_social_benefits.
recipient_social_benefits :-
    renta_minima_insercion.
recipient_social_benefits :-
    ingreso_minimo_vital.

met_common_requirement :-
    disability_status.
disability_status :-
    disabled_parent.
disability_status :-
    disabled_sibling.
%%  Specific requirements:
met_specific_requirement :-
    sibling_enroll_center.
met_specific_requirement :-
    legal_guardian_work_center.

met_specific_requirement :-
    relative_former_student.

met_specific_requirement :-
    school_proximity.
school_proximity :-
    same_education_district.
school_proximity :-
    not same_education_district,
    force_majeure.    % Ambiguity
force_majeure :-
    not n_force_majeure.
n_force_majeure :-
    not force_majeure.
%% Exceptions:
exception :-
    come_non_bilingual,
    want_bilingual_section(Course),
    not accredit_english_level(Course).
accredit_english_level('1st ESO') :-
    b1_certificate.
accredit_english_level('2nd ESO') :-
    b1_certificate.
accredit_english_level('3rd ESO') :-
    b2_certificate.
accredit_english_level('4th ESO') :-
    b2_certificate.
%% Discretion To Act:
obtain_place :-
    not met_requirement,
    met_complementary_criterion(CC).
obtain_place :-
    met_requirement, exception,
    met_complementary_criterion(CC).

met_complementary_criterion(CC) :-
    school_criteria(CC),
    purpose(CC), not unlawful(CC),
    not n_met_complementary_criterion(CC).
n_nmet_complementary_criterion(CC) :-
    not met_complementary_criterion(CC).
purpose(CC) :-
    promote_diversity(CC).
unlawful(CC) :-
    sex_discrimination(CC).
unlawful(CC) :-
    race_discrimination(CC).
unlawful(CC) :-
    religion_discrimination(CC).

school_criteria(foreign_student) :-
    foreign_student.
school_criteria(specific_etnia) :-
    specific_etnia.

promote_diversity(foreign_student).
promote_diversity(specific_etnia).
race_discrimination(specific_etnia).
```

Fig. 1: Translation of the procedure for awarding school places under s(LAW).

example: "Students coming from non-bilingual public schools, who apply for a place in English language bilingual schools and who wish to study in the Bilingual Section, need to accredit a level of English in the four skills equivalent to level B1 for $1^{st}/2^{nd}$ ESO, and to level B2 for $3^{rd}/4^{th}$ ESO".

Ambiguity Ambiguity occurs when some aspects of the law can be interpreted in different ways. For example, "proximity to the family or work address" is a specific and defined requirement based on the distribution by educational districts. However, in case of *force majeure*, students from a education district may be reassigned to a school from another district. Fig. 1 lines 34-44 encode this scenario allowing evaluation without having to determine a priori the force majeure circumstances necessary to justify the reassignment of students. This pattern generates a model where `force_majeure` is assumed to hold and another model where there is *no* evidence that `force_majeure` holds.

Discretion To Act The discretion to act introduces different possible interpretations of the law and/or the contract that we intent to model by generating multiple models. Implementations based on Prolog compute a single, canonical model, and therefore, bypass this non determinism by selecting one interpretation. The discretion to act can be considered as a ground or an exception following the previous patterns. For example, Fig. 1 lines 59-79 shows the translation of the discretion to act rule: "The School Council may add another complementary criterion". The resulting encoding uses predicates in which the variable `CC` can be instantiated with different values. This feature allows us to reuse some of the clauses without repeating them, i.e., the clauses in lines 59-79 are generic, while clauses 81-88 specify the ground and exceptions of the criteria added by a particular school. Clauses in lines 66-71 generate two possible models if the discretion to act is exercised according to the purpose / intention of the law and it is not unlawful. In one model the complementary criterion is applied and in the other it does not. Then, clauses in lines 86-88 state the cases in which the discretion to act has a purpose and/or is unlawful.

Unknown Information The use of default negation may introduce unexpected results in the absence of information (positive and/or negative). Therefore, in many cases the desirable behavior should capture the absence of information by generating different models depending on the relevant information. For example, it may be unclear whether the documents we have to certify that we are a `large_family` are valid or not, so we avoid introducing that information and the reasoner would reason assuming both scenarios. To state that some information is certain we would use the predicate `evidence/1`, e.g., `evidence(large_family)` means that s/he has the condition of large family. Additionally, s(LAW) would provide *strong* negation, denoted with `'-'`, to specify that we have evidences supporting the falsehood of some information, e.g., `-evidence(large_family)` means that s/he does not have the condition of large family.

3.2 Description of s(LAW)

s(LAW), built on top of s(CASP), is composed by three modules: the first contains the *articles*, the second contains *explanations* to generate readable justifications, and the third one contains the *evidences*. In our running example:

```
s/he may obtain a school place, because
    a common requirement is met, because
        s/he is part of a large family.
    a specific requirement is met, because
        s/he has siblings enrolled in the center.
    there is no evidence that an exception applies, because
        s/he came from a non-bilingual public school, and
        s/he wish to study 2nd ESO in the Bilingual Section, and
        s/he accredit required level of English for 2nd ESO, because
            in the four skills certificate level b1.
```

Fig. 3: Justification in Natural Language for the evaluation of `student01.pl`.

ArticleESO.pl Contains the legislation rules in Fig. 1 following the patterns described in Section 3.1.

ArticleESO.pre.pl Contains the natural language patterns for the predicates that are relevant to provide readable justifications of the conclusions inferred by s(LAW). The directive `#pred` defines the natural language patterns, e.g.:

```
#pred obtain_place :: 's/he may obtain a school place'.
```

Additionally, to facilitate the understanding of the code we can obtain a readable code (in natural language) by invoking `scasp --code --human`.

StudentXX.pl Fig. 2 shows the encoding of the module `student01.pl` corresponding to one student. This last module encodes the evidences of a student and links them with the previous modules `ArticleESO.pl` and `ArticleESO.pred.pl` (lines 1-2). The predicates `evidences/1` and `-evidence/1` (explained in Section 3.1) are used to specify the known information (positive or negative evidences). For the sake of brevity, let us handle as *unknown* the evidences corresponding to: `large_family`, `renta_minima_insercion`, `sibling_enroll_center`, `same_education_district`, `b1_certificate`, `foreign_student`, and `specific_etnia`. Fig. 2 lines 7-13 provide the known information corresponding to this student. Additionally, we consider that the students, coming from non-bilingual public schools, apply for a place in English language bilingual schools and wish to study in the Bilingual Section (Fig. 2 lines 4-5).

```
1  #include('ArticleESO.pl').
2  #include('ArticleESO.pred.pl').
3
4  come_non_bilingual.
5  want_bilingual_section('2nd ESO').
6
7  evidence(large_family).
8  evidence(renta_minima_insercion).
9  evidence(sibling_enroll_center).
10 evidence(same_education_district).
11 evidence(b1_certificate).
12 -evidence(foreign_student).
13 -evidence(specific_etnia).
```

Fig. 2: File `student01.pl`.

4 Reasoning and Deduction with Real Use-Cases

The modules of s(LAW) are implemented under s(CASP) version 0.21.04.04 (https://gitlab.software.imdea.org/ciao-lang/sCASP), that runs under Ciao

Table 1: Case of different students evaluated using s(LAW).
Note: ‘+’ is a positive evidence, ‘−’ is a negative evidence, ‘?’ means unkown.

	st_1	st_2	st_3	st_4	st_5	st_6
`large_family`	+	+	+	−	−	−
`renta_minima_insercion`	+	+	+	?	−	−
`sibling_enroll_center`	+	+	−	+	−	−
`same_education_district`	+	+	−	+	−	−
`b1_certificate`	+	−	+	?	−	−
`foreign_student`	−	−	−	−	+	−
`specific_etnia`	−	−	−	−	−	+
`?- obtain_place`	**yes**	no	**yes**	**yes**	**yes**	no

Prolog version 1.19-480. (http://ciao-lang.org/). The benchmarks used in this section are available at http://platon.etsii.urjc.es/~jarias/papers/slaw-caepia21 and were run on a MacOS 11.2.3 laptop with an Intel Core i7 at 2.6 GHz.

A priori Deduction: Consider we run our reasoner s(LAW) in the interactive mode to reason about six different students by invoking:

```
scasp -i --tree --human --short studentXX.pl
```

where XX corresponds to the ‘id’ of each student (from 1 to 6). Then, we ask the queries to obtain conclusions from the reasoner. Table 1 shows the data corresponding to the candidates and the conclusion generated by s(LAW) for the query `?- obtain_place`. Students 1, 3, 4, and 5 obtain a place at the school while students 2 and 6 do not.

– Student 1: Fig. 2 contains the information corresponding to this student. Since s/he meets common and specific requirements and avoids the exception (having level b1 in English), the evaluation returns the partial model:

```
{ obtain_place, large_family, sibling_enroll_center, come_non_bilingual,
  want_bilingual_section(2nd ESO), b1_certificate }
```

and the corresponding justification shown in Fig. 3.
– Student 2: meets common and specific requirements but has to be rejected because s/he does not accredit level b1 in English.
– Student 3: meets common requirements, avoids the exception and by assuming `force_majeure` s/he also meets a specific requirement (school proximity). Note that s/he does not live in the same education district.
– Student 4: in this use-case there is absence of information regarding the “renta minima de insercion” and the English certificate (marked with ?). The partial model returned assumes that the truth values for these pieces of information are true. Therefore, based on that assumption the student would obtain a place.

```
 1 there is no evidence that s/he may obtain a school place, because
 2     there is no evidence that a common requirement is met, because
 3         there is no evidence that s/he is part of a large family, and
 4         there is no evidence that s/he is a recipient of the RMI, and
 5         there is no evidence that a parent or sibling has disability status.
 6     there is no evidence that the criterion foreign_student is met, because
 7         there is no evidence that s/he meets the criteria foreign_student, because
 8             there is no evidence that s/he is a foreign student.
 9     there is no evidence that the criterion specific_etnia is met, because
10         s/he meets the criteria specific_etnia, because
11             s/he belongs to a specific etnia.
12         specific_etnia follows the purpose of the procedure, because
13             specific_etnia promotes the diversity.
14         specific_etnia is illegal, because
15             specific_etnia discriminates based on race.
```

Fig. 4: Justification in Natural Language for the evaluation of `student06.pl`.

- Student 5: now we consider that the school added a complementary criterion for foreign students and therefore, since the student is a foreigner, s/he obtains a place.
- Student 6: in this use-case the complementary criterion `specific_etnia` cannot be applied because it discriminates by race and therefore, it is unlawful. Therefore, the student does not obtain a place.

A posteriori Deduction The main advantage of s(LAW) is its ability to generate justifications not only for positive but also for negative information. This ability allows us to analyze the reason for a specific inference and/or to determine which are the requirements needed to obtain a specific conclusion:

- For student 3, the query `?- not force_majeure, obtain_place` avoids the assumption of force majeure and the student does not obtain a place.
- For student 4, the query `?- not obtain_place` returns the partial models (with the assumptions) for which this student does not obtain a place.
- For student 6, Fig. 4 shows the justification of the query `?- not obtain_place` so we can analyze more in detail why this student is rejected. While the complementary criteria for student 5 (`foreign_student`) is similar to `specific_etnia`, the justification tree shows that this student does not obtain a place because the complementary criterion is illegal (Fig. 4 lines 14-15).

Additionally, we can collect the partial models, in which the school place is or is not obtained, together with their justification and analyze "Epistemic Specifications" [6], that is, what is true in all/some models, which partial models share certain assumptions, etc. This reasoning makes it possible to detect the missing information that would change the decision from "not obtained" (or "obtained" under some assumptions) to "obtained". Note that, by introducing the new evidences, the resulting justification of s(LAW) provides an explanation in which these evidences are used to support the decision.

5 Related Work

Most ASP systems follow bottom-up executions that require a grounding phase where the variables of the program are replaced with their possible values. During the grounding phase, links between variables are lost and therefore an explanation framework for these systems must face many challenges to provide a concise justification of why a specific answer set satisfies the rules (and which rules). The most relevant approaches are: off-line and on-line justifications [11]; Causal Graph Justification [2]; and Labeled ABA-Based Answer Set Justification (LABAS) [13]. However, these approaches are applied to grounded versions of the programs, i.e., non-ground programs have to be grounded, and they may produce unwieldy justifications when the non-ground program has uninterpreted functions, consults large databases and/or requires the representation of dense domains [1].

On the other hand, systems that follow a top-down execution can trace which rules have been used to obtain the answers more easily. One such system is ErgoAI (https://coherentknowledge.com), based on XSB [16], that generates justification trees for programs with variables. ErgoAI has been applied to analyze streams of financial regulatory and policy compliance in near real-time providing explanations in English that are fully detailed and interactively navigable. However, default negation in ErgoAI is based on the well-founded semantics [5] and therefore ErgoAI is not a framework that allows the representation of ambiguity and/or administrative discretion.

Finally, we would like to emphasize that explainable AI techniques for black-box AI tools, most of them based on machine learning, are not able to explain how variation in the input data changes the resulting decision [4].

6 Conclusions

In this paper we have shown that using goal-directed answer set programming, s(LAW) is capable of modeling discretion and ambiguity. The deduction based on s(LAW) allows: the consideration of different conclusions (multiple models) which can be analyzed by humans thanks to the justification generated in natural language; and the reasoning about the set of these conclusions/models. To the best of our knowledge, s(LAW) is the only system that exhibits the property of modelling vague concepts.[3]

Our future work unfolds among two major lines. The first is to complete the modeling of the legislation by tabulation for each of the criteria used in the procedure for adjudication of school places in centers supported with public funds. And, second, the use of this tabulation of criteria to check (by employing the underlying constraint solver of s(SCASP)) whether automated decisions can be made when the regulation includes ambiguity, administrative discretion and unknown information.

[3] On January 14^{th}, 2021, Dr. Robert Kowalski explained how they bypassed in [14] the representation of vague concepts such as *without undue delay* [8, 1:20:15, 1:26:00].

References

1. Arias, J., Carro, M., Salazar, E., Marple, K., Gupta, G.: Constraint Answer Set Programming without Grounding. Theory and Practice of Logic Programming **18**(3-4), 337–354 (2018). https://doi.org/10.1017/S1471068418000285
2. Cabalar, P., Fandinno, J., Fink, M.: Causal Graph Justifications of Logic Programs. Theory and Practice of Logic Programming **14**(4-5), 603–618 (2014). https://doi.org/10.1017/S1471068414000234
3. Cobbe, J.: Administrative law and the machines of government: judicial review of automated public-sector decision-making. Legal Studies **39**(4), 636–655 (2019)
4. DARPA: Explainable Artificial Intelligence (XAI). Defense Advanced Research Projects Agency (2017), https://www.darpa.mil/program/explainable-artificial-intelligence
5. Gelder, A.V., Ross, K., Schlipf, J.: The Well-Founded Semantics for General Logic Programs. Journal of the ACM **38**, 620–650 (1991). https://doi.org/10.1145/116825.116838
6. Gelfond, M.: Logic programming and reasoning with incomplete information. Annals of mathematics and artificial intelligence **12**(1), 89–116 (1994)
7. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: 5th International Conference on Logic Programming. pp. 1070–1080 (1988), http://www.cse.unsw.edu.au/~cs4415/2010/resources/stable.pdf
8. Kowalski, R.A.: Logical English = Logic + English + Compupting. https://utdallas.app.box.com/s/ngsyloscj5sk24uh3axexxz451o74z0u (January 2021), HackReason Opening Ceremony. Last accessed 19 April 2021
9. Marple, K., Salazar, E., Gupta, G.: Computing Stable Models of Normal Logic Programs Without Grounding. arXiv **1709.00501** (2017), http://arxiv.org/abs/1709.00501
10. Cerrillo i Martínez, A.: El derecho para una inteligencia artificial centrada en el ser humano y al servicio de las instituciones: Presentación del monográfico. IDP: Revista de Internet, Derecho y Politica (30) (2019)
11. Pontelli, E., Son, T.C., El-Khatib, O.: Justifications for Logic Programs under Answer Set Semantics. Theory and Practice of Logic Programming **9**(1), 1–56 (2009). https://doi.org/10.1017/S1471068408003633
12. Ramakrishna, S., Górski, Ł., Paschke, A.: A dialogue between a lawyer and computer scientist: the evaluation of knowledge transformation from legal text to computer-readable format. Applied Artificial Intelligence **30**(3), 216–232 (2016)
13. Schulz, C., Toni, F.: Justifying Answer Sets Using Argumentation. Theory and Practice of Logic Programming **16**(1), 59–110 (2016). https://doi.org/10.1017/S1471068414000702
14. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Communications of the ACM **29**(5), 370–386 (1986)
15. Solé, J.P.: Inteligencia artificial, derecho administrativo y reserva de humanidad: algoritmos y procedimiento administrativo debido tecnológico. Revista general de Derecho administrativo **50** (2019)
16. Swift, T., Warren, D.S.: XSB: Extending Prolog with Tabled Logic Programming. Theory and Practice of Logic Programming **12**(1-2), 157–187 (Jan 2012). https://doi.org/10.1017/S1471068411000500