Integration of Logical English and s(CASP)

Galileo Sartor[1], Jacinto Dávila[2], Marco Billi[3], Giuseppe Contissa[3], Giuseppe Pisano[3] and Robert Kowalski[4]

[1]*Department of Computing, University of Turin, Turin*
[2]*Contratos Lógicos. C.A. and Universidad de Los Andes Mérida, Venezuela*
[3]*Department of Law, University of Bologna, Bologna*
[4]*Department of Computing, Imperial College, London, UK*

Abstract
This paper demonstrates the use of Logical English as a logic programming language that can be interpreted by the s(CASP) reasoner. It shows how legal knowledge and unknown information can be expressed in a form of English that can be easily understood by users without any technical training and with only a basic knowledge of English. This research has been undertaken in the context of the CrossJustice Project.

Keywords
Logic Programming, Prolog, Controlled Natural Language, Legal Rule Modelling, Explainable AI, Logical English

1. Introduction

The goal of this paper is to explore the use of Logical English (LE) and s(CASP) in the CrossJustice system, and to assess their use both for drafting legal norms in logical form and for making the system more accessible for users who lack a technical background in computing or symbolic logic.

The focus of the CrossJustice Project[1] is to develop a platform for normative harmonization assessment and automated legal support for European criminal procedural rights of suspected and accused persons. In the following sections we will analyse a short example from the CrossJustice knowledge base, rewritten in LE, and interpreted by s(CASP). We will show how to draft rules in LE and how these rules are converted into s(CASP), to take advantage of the explanation system of s(CASP). We will explore the utility for the legal domain of the ability in s(CASP) to reason with unknown information, represented by abductive (or assumable) predicates and restricted by the use of constraints.

2nd Workshop on Goal-directed Execution of Answer Set Programs (GDE'22), August 1, 2022
galileo.sartor@unito.it (G. Sartor); jd@logicalcontracts.com (J. Dávila); marco.billi3@unibo.it (M. Billi); giuseppe.contissa@unibo.it (G. Contissa); g.pisano@unibo.it (G. Pisano); r.kowalski@imperial.ac.uk (R. Kowalski)
0000-0001-6355-851X (G. Sartor); 0000-0003-1053-8763 (J. Dávila); 0000-0002-6807-073X (M. Billi); 0000-0002-8511-1505 (G. Contissa); 0000-0000-0000-0000 (G. Pisano); 0000-0002-1341-8583 (R. Kowalski)

2. Background

Logical English (LE)[2, 3, 4] is a general-purpose computer language[1], which is designed to be efficiently executed by computer, yet easily understandable by a reader of English without any technical training. Programs and other kinds of knowledge are expressed in LE in the form of facts and rules of the logical form *conclusion if conditions*, and they are executed by translating them into a logic programming language such Prolog or s(CASP).

An LE document consists of a knowledge base of facts and rules, scenarios, queries and templates. The templates are declarations of the predicates contained in the knowledge base and scenarios, such as

**a person* has *a right* according to *an article*.*

A template identifies a predicate and its variable argument places. The argument places are identified by a simple noun phrase surrounded by asterisks and starting with an indefinite article "a" or "an". The predicate itself is represented by the rest of the template. The templates are used to identify instances of the predicates in the knowledge base and elsewhere. For example, the template above can be used to identify a sentence such as:

Mario has the right to interpretation according to article 3.

as an instance of a predicate, which is translated into Prolog or s(CASP) resulting in the symbolic representation:

has_according_to('Mario', 'right_to_interpretation', 'Article_3').

The translation can be processed by any Prolog or s(CASP) implementation. We use a SWI-Prolog SWISH Web Platform with support for LE and s(CASP) for this processing.

Logical English can be used to model legal norms in logical form. As argued, for example, by [5], some of the needs for legal modelling are (1) isomorphism, understood as a correspondence between the legal source and the knowledge base, preserving possible interpretations, references and connections, and (2) defeasibility, which enables reasoning with rules and exceptions, where the effect of a legal rule may be blocked by the applicability of another rule.

In this paper we focus on the integration of LE and s(CASP) [6], which builds on Answer Set Programming (ASP), a well-known logic programming paradigm used to solve NP-hard problems. s(CASP) executes ASP using an efficient top-down, SLD-like resolution procedure, while also incorporating the generation of assumptions (i.e. abduction) and constraints on which assumptions can be generated.

The combination of LE and s(CASP) seems to be mutually beneficial. For example, the common nouns used in the names of variables in LE can be used to assign types to arguments of predicates in s(CASP) explanations. In the above example this would result in s(CASP) explanations containing an explicit reference to: *the person Mario.*

Recent developments have also introduced initial support for multiple languages in LE. Work on this multi-language support can be a common effort useful to both LE and s(CASP).

[1]https://github.com/LogicalContracts/LogicalEnglish/

3. The CrossJustice Example in Logical English

The CrossJustice Project is an expert system containing a knowledge base about European and national laws relating to the rights granted to suspect or accused persons. The system is now available for public use [7]. The user of the system completes a form stating the relevant facts of their case, and is then shown an answer, with references to the relevant legal texts. In the CrossJustice project, all the legal norms have been encoded in Prolog, which has proved to be well-suited to the requirements of the legal context, and which can be enhanced with additional features[8] implemented in Prolog, such as the LE interface described in this paper.

The running example in this paper focuses on Articles 3(1), 3(2) and 3(3) of Directive 2010/64. Article 3(1) states that suspected or accused persons are to be provided with a written translation of all essential documents in a legal case, if they do not speak the language of the proceedings. Articles 3(2) and 3(3) give examples of what constitutes an essential document, such as a decision depriving a person of their liberty, a charge or indictment, or a judgement. In addition, Article 3(3) states that the court has the power to decree that any document may be considered essential for the defence of the person. This open concept[9] of an *essential document* is left vague on purpose, giving the courts the discretion to interpret the same concept differently within the different legal traditions of the different EU member States.

In our example, we have used the English language to translate the above-mentioned articles from the European directive into Logical English. However, LE potentially supports the drafting of programs in multiple natural languages. The goal of the following LE rule is to verify whether a person, such as Mario, has the right to translation of essential documents, according to Article 3.

Listing 1: Does a person have a right to translation?

```
a person has the right to translation of an essential document according to Article 3_1
    if the language of the proceedings of the person is a given language
    and a document needed by the person is recognised as an essential document
    and it is not the case that
        the person understands the given language.
```

The LE knowledge base also includes rules representing the the open list of essential document types, which are stated explicitly in the legal text.

In the interest of facilitating the translation from LE into s(CASP), we have expressed each essential document type by means of a separate rule, although in LE it is also possible to express the same information with a single rule, with each condition listed as an alternative using disjunction.

We have also implemented a rule, representing Article 1 of the Directive, which states that all rights referred to in the Directive shall apply to persons who are made aware by the competent authorities of a Member State, that they are suspected or accused of having committed a criminal offence, from the start of the proceedings until its conclusion.

In order to reason with the times referred to in this rule, we have added an additional rule representing the event calculus for reasoning about the relationship between the time of an event and the times of the fluents (i.e. time-varying facts) that start or end upon the occurrence of the event.

It is clear from these examples of rules written in LE that, although still verbose, the use of natural language connectors (*if*, *and*, *or*) and the use of the same words in LE templates as in the legal source can help a person with no background in computer programming to understand the LE rules and thus to understand the legal conditions for the applicability of the rules. This understanding is closely related to what we referred to previously as an isomorphic representation of knowledge.

Finally, an LE document contains one or more scenarios and one or more queries. For example:

Listing 2: A case scenario

```
1  scenario one is:
2  the language of the proceedings of Mario is dutch.
3  the document D1 needed by Mario concerns charge.
4  Mario is involved in proceedings of criminal.
5  the proceedings have started at 2000-05-13.
6  Mario has been made aware that Mario has the status of accused.
7  it is unknown whether the authority has decided that
8      the document D1 needed by Mario concerns essential document.
9  the proceedings have ended at 2001-01-08.
10
11 query one is:
12 which fluent holds at 2000-07-01.
```

The facts on lines 5 and 9 are used together with the event calculus to derive that the proceedings take place at every time between the start at $2000-05-13$ and the end at $2001-01-08$.

In an LE document the goal or goals are defined in a specific query section, and in this case the goal is to verify whether the user has the right to translation on a date, $2000-07-01$, which comes after the start of the proceedings and before the proceedings have ended. In the query the variables are identified by keywords such as *which* (as well as by *a* and *an*), and they are instantiated in the answer.

3.1. s(CASP) translation

In this section we remark upon some of the issues that arise concerning the translation of rules from LE into s(CASP). For example, the s(CASP) rules generated by this translation can be seen in listing 3.

Listing 3: Translation into Prolog

```
has_according_to(A, the_right_to_translation_of_an_essential_document, 'Article_3_1')
    :-
    the_language_of_the_proceedings_of_is(A, B),
    needed_by_is_recognised_as_an_essential_document(_, A),
    not understands(A, B).
```

Here negation in LE is translated into negation by failure in s(CASP), which derives that a condition does not hold if it cannot be shown that the condition does hold. In addition to negation by failure, s(CASP) also supports reasoning with an explicit form of negation which expresses that the opposite (or contrary) of a predicate does hold (i.e., so-called strong negation)[10]. The combination of the two kinds of negations makes it possible to express such expressions as "the contrary of a condition does not hold".

s(CASP) generates explanation trees in English using a directive *#pred* in the s(CASP) program to map predicates into readable English sentences. The translation of LE into s(CASP) translates the LE templates into these *#pred* directives.

One of the most interesting features of s(CASP) for legal reasoning is the possibility of defining predicates as unknown (or "abducible"), which allows the user to state that a certain fact is neither certainly true nor certainly false[11]. The s(CASP) reasoner can solve a query by using assumptions, which are instances of these abducible predicates. In this example it is unknown whether the authority has deemed the document as essential in the proceeding.

Listing 4: Handling unknowns

```
it is unknown whether the authority has decided that
    the document D1 needed by Mario concerns essential document.
```

The LE to s(CASP) translator produces the *abducible* statement of listing 5.

Listing 5: Translation into sCASP

```
#abducible the_authority_has_decided_that(needed_by_is_of_type('D1', 'Mario',
    essential_document)).
```

During the process of query evaluation, when the s(CASP) reasoner encounters an abducible condition as a subgoal, it checks whether the condition is explicitly negated by a constraint in the knowledge base. If it is not explicitly negated, then the reasoning continues, both under the assumption that the abducible subgoal is true, and under the alternative assumption that the abducible subgoal is not true.

3.1.1. Possible issues in the s(CASP) translation

In defining the translation from LE to s(CASP), we have encountered a few limitations, which can be overcome by extending the s(CASP) parser and reasoner.

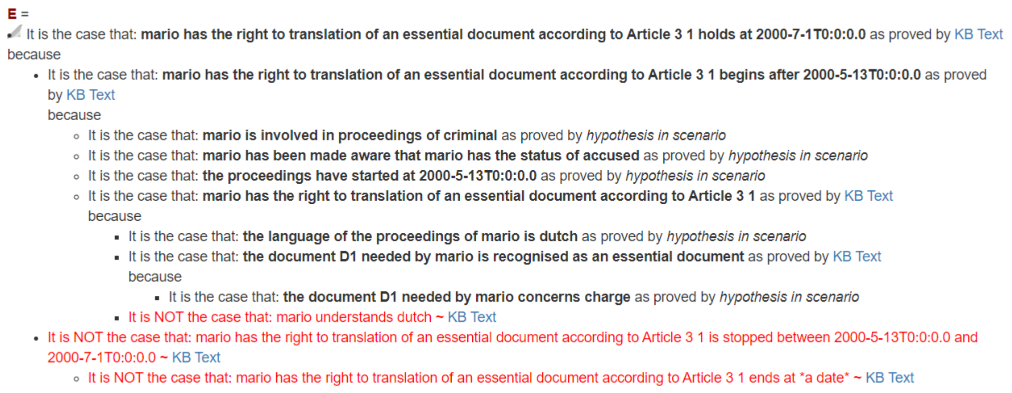
The first limitation of s(CASP) is that it admits only one negated literal in a negative condition. In contrast, in Prolog it is possible to negate a conjunction (or disjunction) of literals. This limitation is most likely derived from the necessity to easily generate duals for all the predicates defined in the knowledge base, which is the way s(CASP) implements constructive negation.

The second related limitation is that s(CASP) does not support disjunction in the body of rules. The use of disjunction in the body of rules can be avoided in general, by writing instead multiple rules with the same head, as we do in the definition of "essential document" in the running example. It would be useful and closer (more isomorphic) to the original legal text to allow disjunctions in the body of rules. But this could complicate the generation of dual predicates.

The third limitation is that explanations in s(CASP) do not translate meta-predicates, which are predicates that have other predicates as arguments, into natural language. For example, the event calculus employs a meta-predicate to express that a fluent holds at a time, where the fluent itself is another predicate (in this case a predicate expressing that a person has the right to translation), and this fluent is not translated into English in s(CASP) explanations.

4. Results and Justification

Finally, we compare explanations in LE with explanations in s(CASP). In both cases, as in other rule-based systems, explanations display the rules used in solving a goal during the course of a problem-solving session, and they are one of the most important requirements for expert AI systems in the legal domain.



E =
It is the case that: **mario has the right to translation of an essential document according to Article 3 1 holds at 2000-7-1T0:0:0.0** as proved by KB Text
because

- It is the case that: **mario has the right to translation of an essential document according to Article 3 1 begins after 2000-5-13T0:0:0.0** as proved by KB Text
  because
  - It is the case that: **mario is involved in proceedings of criminal** as proved by *hypothesis in scenario*
  - It is the case that: **mario has been made aware that mario has the status of accused** as proved by *hypothesis in scenario*
  - It is the case that: **the proceedings have started at 2000-5-13T0:0:0.0** as proved by *hypothesis in scenario*
  - It is the case that: **mario has the right to translation of an essential document according to Article 3 1** as proved by KB Text
    because
    - It is the case that: **the language of the proceedings of mario is dutch** as proved by *hypothesis in scenario*
    - It is the case that: **the document D1 needed by mario is recognised as an essential document** as proved by KB Text
      because
      - It is the case that: **the document D1 needed by mario concerns charge** as proved by *hypothesis in scenario*
    - It is NOT the case that: mario understands dutch ~ KB Text
- It is NOT the case that: mario has the right to translation of an essential document according to Article 3 1 is stopped between 2000-5-13T0:0:0.0 and 2000-7-1T0:0:0.0 ~ KB Text
  - It is NOT the case that: mario has the right to translation of an essential document according to Article 3 1 ends at *a date* ~ KB Text

Figure 1: Logical English explanation

Figures 1 and 2 illustrate different explanations for the same goal of showing that Mario has the right to translation of an essential document at a certain date.

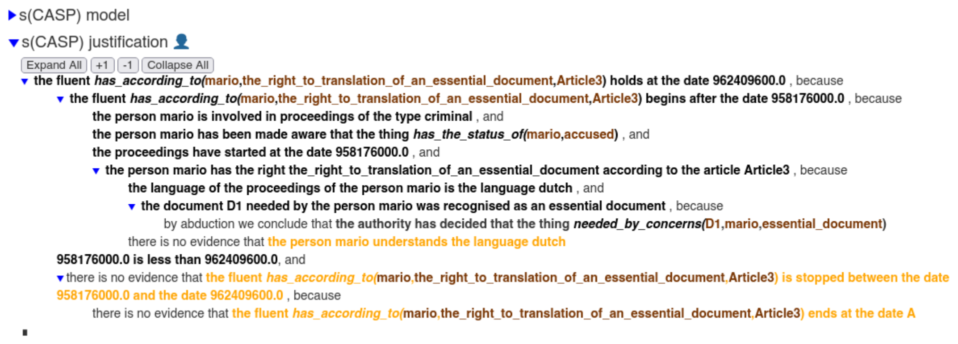
In both explanations, 1 and 2, the goal (or conclusion) is presented at the top, and the conditions that justify the goal are presented underneath the goal. Both explanations include among these justifying conditions the negative condition that, in the given scenario, it cannot be shown that Mario understands dutch.

There is only one explanation, which is shown in figure 1, associated with the translation from LE into Prolog. This explanation uses the non-abductive "hypothesis" in the given scenario that the document D1 needed by Mario concerns a charge, which according to Article 3(2) is one of the types of document explicitly recognised as essential by the European legislation.

The LE/Prolog exlanation has links (on the right of the conditions in the explanation with the label "KB") to point to the rules of the program that justify deriving that the conditions hold. They also have links (using the label "Text") to point to the original source text on the web. Such features, linking the LE representation with the original source can be very useful both for the legal expert and for the non-expert user.

In contrast with the one explanation in Prolog, there are two possible explanations for the same goal in s(CASP). One of the s(CASP) explanations corresponds to the Prolog explanation without abduction, and uses the same facts given in the scenario. The other explanation, in 2, uses abduction to assume that the judge exercises the discretionary power, expressed in Article 3(3), to deem that document D1 is essential to the defence of Mario. The use of abduction is highlighted in green.

Because meta-predicates are not translated into natural language in s(CASP), the resulting s(CASP) explanation is not easy for an untrained user to read.



```
▾ the fluent has_according_to(mario,the_right_to_translation_of_an_essential_document,Article3) holds at the date 962409600.0 , because
    ▾ the fluent has_according_to(mario,the_right_to_translation_of_an_essential_document,Article3) begins after the date 958176000.0 , because
        the person mario is involved in proceedings of the type criminal , and
        the person mario has been made aware that the thing has_the_status_of(mario,accused) , and
        the proceedings have started at the date 958176000.0 , and
        ▾ the person mario has the right the_right_to_translation_of_an_essential_document according to the article Article3 , because
            the language of the proceedings of the person mario is the language dutch , and
            ▾ the document D1 needed by the person mario was recognised as an essential document , because
                by abduction we conclude that the authority has decided that the thing needed_by_concerns(D1,mario,essential_document)
            there is no evidence that the person mario understands the language dutch
    958176000.0 is less than 962409600.0, and
    ▾there is no evidence that the fluent has_according_to(mario,the_right_to_translation_of_an_essential_document,Article3) is stopped between the date
    958176000.0 and the date 962409600.0 , because
        there is no evidence that the fluent has_according_to(mario,the_right_to_translation_of_an_essential_document,Article3) ends at the date A
▪
```

Figure 2: s(CASP) explanation

5. Conclusions

The open source nature of both LE and s(CASP) facilitates a cross-fertilization of ideas and approaches of the kind presented in this paper.

On the one hand, certain features of Prolog (and thus LE) are not yet included in s(CASP), such as the inclusion of disjunction in the body of the rules. Although it is possible to represent complex rules without the use of disjunction, the inclusion of disjunction would enhance the ability of s(CASP) to represent legal texts isomorphically.

On the other hand, the ability of s(CASP) to reason abductively with predicates whose truth values are unknown, is very useful in legal reasoning, especially when considering the multiple sources of law and their validity relative to one another. Future work extending the translation from LE to Prolog, to include abduction, may prove fruitful, especially when combined with extensions to include reasoning by means of argumentation.

6. Acknowledgements

We are grateful to Jan Wielemaker for adapting the original s(CASP) system to run on the SWISH platform[2], enabling us to do the work presented in this paper.

This work has been supported by the H2020 ERC Project "CompuLaw" (G.A. 833647), and 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".

[2]https://github.com/JanWielemaker/sCASP

References

[1] M. Billi, R. Calegari, G. Contissa, G. Pisano, G. Sartor, G. Sartor, Explainability through argumentation in logic programming, in: CAUSAL'21: Workshop on Causal Reasoning and Explanation in Logic Programming, 2021. URL: http://ceur-ws.org/Vol-2970/causalpaper2.pdf.

[2] R. Kowalski, Logical english, Proceedings of Logic and Practice of Programming (LPOP) (2020).

[3] R. Kowalski, A. Datoo, Logical english meets legal english for swaps and derivatives, Artificial Intelligence and Law 30 (2022) 163–197.

[4] R. Kowalski, J. Dávila, C. L. CA, M. Calejo, Logical english for legal applications, Conference: XAIF, Virtual Workshop on XAI in Finance (2021).

[5] T. F. Gordon, G. Governatori, A. Rotolo, Rules and norms: Requirements for rule interchange languages in the legal domain, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 282–296. doi:10.1007/978-3-642-04985-9_26.

[6] J. Arias, M. Carro, E. Salazar, K. Marple, G. Gupta, Constraint answer set programming without grounding, Theory and Practice of Logic Programming 18 (2018) 337–354. doi:10.1017/S1471068418000285.

[7] Crossjustice platform, https://legalmachinelab.unibo.it/crossjustice, 2022.

[8] G. Contissa, G. Sartor, Legal knowledge representation in the domain of private international law?, volume 2781 of *CEUR Workshop Proceedings*, CEUR, Venice, Italy, 2020, pp. 83–92. URL: http://ceur-ws.org/Vol-2781/#paper8.

[9] H. L. A. Hart, H. L. A. Hart, J. Raz, L. Green, The Concept of Law, Oxford University Press, 2012.

[10] S. C. Guy, R. Schwitter, The PENG ASP system: architecture, language and authoring tool, Language Resources and Evaluation 51 (2016) 67–92. doi:10.1007/s10579-016-9338-7.

[11] J. Morris, Constraint Answer Set Programming as a Tool to Improve Legislative Drafting: A Rules as Code Experiment, Association for Computing Machinery, New York, NY, USA, 2021, p. 262–263. URL: https://doi.org/10.1145/3462757.3466084.