6-2021

Constraint answer set programming as a tool to improve legislative drafting

Jason MORRIS
Singapore Management University, jmorris@smu.edu.sg

Citation

MORRIS, Jason. Constraint answer set programming as a tool to improve legislative drafting. (2021). *Proceedings of the 18th International Conference on Artificial Intelligence and Law, São Paulo, Brazil, 2021 June 21-25*. 262-263.
Available at: https://ink.library.smu.edu.sg/cclaw/7

Constraint Answer Set Programming as a Tool to Improve Legislative Drafting

A Rules as Code Experiment

Jason Morris
jmorris@smu.edu.sg
Singapore Management University Centre for Computational Law
Singapore

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

legal knowledge representation and reasoning, constraint answer set programming, rules as code

ACM Reference Format:
Jason Morris. 2021. Constraint Answer Set Programming as a Tool to Improve Legislative Drafting: A Rules as Code Experiment. In *Eighteenth International Conference for Artificial Intelligence and Law (ICAIL'21), June 21–25, 2021, São Paulo, Brazil.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3462757.3466084

1 RULES AS CODE

"Rules as Code" in this paper is used to refer to a proposed methodology of legislative and regulatory drafting.[1] That legislation can be represented in declarative code for automation has long been recognized [6], as has the opportunity for improving the quality of legal drafting with the techniques of formal representation [1].

Rules as Code further proposes that both drafting and automation would be improved by initially co-drafting statute law in both natural and computer languages simultaneously [4].

Knowledge acquisition bottlenecks and roadblocks associated with statutory interpretation are largely avoided. The co-drafted encoding need only reflect what the legislation says, and not what the legislators meant. Legislative intent is instead encoded as tests by people with authoritative knowledge of the intent, the drafters. In this way, failed tests can be used in the drafting process to signal issues with the natural language draft. When the drafting process is complete an authoritative encoding consistent with the legislative intent already exist. This encoding can be used by regulators and regulated entities to automate services and compliance tasks.

[1] The phrase "Rules as Code" is also often used to refer to legal knowledge representation and reasoning generally.

ICAIL'21, June 21–25, 2021, São Paulo, Brazil

ACM ISBN 978-1-4503-8526-8/21/06.
https://doi.org/10.1145/3462757.3466084

2 S(CASP)

s(CASP) is a stable-model constraint answer set programming language, implemented in the Ciao programming language [3]. s(CASP) was selected for use in this experiment because of its ability to generate natural language explanations for answer sets [2], and its ability to perform both deductive and abductive reasoning from the same encoding with minor adjustments [3]. Justifications for automated conclusions have long been recognized as useful in legal applications both for end-users and as a development tool [5].

3 RULE 34, LEGAL PROFESSION (PROFESSIONAL CONDUCT) RULES OF SINGAPORE, 2015

Singapore's *Legal Profession Act*, (Cap. 161) (the "*Act*") governs the legal profession in Singapore. Part VI of the *Act* establishes the Professional Conduct Council, which in section 71(2) of the *Act* is given broad authority for drafting rules governing the practice, conduct, etiquette, and discipline of legal practitioners in Singapore. The Professional Conduct Council has enacted the Legal Profession (Professional Conduct) Rules (S 706/2015) ("the Rules"). In 2015 the Rules were significantly amended with a new Rule 34 setting out restrictions on lawyers accepting executive appointments outside of their legal practice.

4 EXPERIMENTAL DESIGN

Our interdisciplinary team undertook to assess the strengths and weaknesses of s(CASP) as a tool for improving legislative drafting in a Rules as Code approach. The author encoded a literal interpretation of Rule 34 in s(CASP), and separately encoded the author's expectations of the behaviour of Rule 34 as a set of tests. Test failures that the author attributed only or primarily to issues with the natural language drafting of Rule 34 were raised with legally-trained team members to confirm whether the expected behaviour was reasonable, and whether the cause of the test failure was a legal drafting issue.

The discovery of such issues would demonstrate the feasibility of using s(CASP) to detect legislative drafting issues. Any issues arising would be in the context of non-authoritative opinions as to the expected behaviour of the text. The experiment cannot, therefore, be used to diagnose issues with the Rule as enacted.

5 EXPERIMENTAL RESULTS

The code written to implement this experiment is available at https://github.com/smucclaw/r34_scasp.

A set of 25 tests were encoded, and there were 4 test failures not explained by errors encoding the Rule or the tests. These four failures were investigated by the author by performing 'why not' queries and reviewing the justifications provided by s(CASP).

This process revealed that the failing tests were encoded on the basis of an expectation that the word "business" in Rule 34(1)(b) referred to a legal practitioner's activities. But Rule 34(9) defines "business" to refer to a general category of undertaking. Setting out a test in which Rule 34(1)(b) applied, while also using the defined meaning of "business", required making statements that did not have clearly meaningful real world equivalents. This suggested that Rule 34(1)(b) might also use the word "business" in a way inconsistent with the defined meaning, which would be a drafting issue.

That issue was raised with the rest of the research team, who confirmed that Rule 34(1)(b) had been faithfully encoded, that the expectations of the failing tests were reasonable, and that Rule 34(1)(b) required the use of an interpretations of the word "business" that is inconsistent with the defined meaning of the word in order to give effect to that expectation, or to give it any clear meaning at all.

The research team seriously considered the possibility that there might be a different interpretation of other aspects of the Rule that would make Rule 34(1)(b) more clearly meaningful. The team was unable to find an interpretation that would have had that effect and would not also make Rule 34(1)(b) redundant to other portions of the Rule. The research team therefore concluded that it would be more correct if Rule 34(1)(b) referred not to businesses but to the holding of an executive appointment.

The researchers agreed on the following proposed replacement for Rule 34(1)(b):

> (1A) A legal practitioner must not accept any executive appointment that materially interferes with —
> (i) the legal practitioner's primary occupation of practising as a lawyer;
> (ii) the legal practitioner's availability to those who may seek the legal practitioner's services as a lawyer; or
> (iii) the representation of the legal practitioner's clients.

The proposed amendment was encoded, and the tests re-run. All 25 tests passed.

6 CONCLUSIONS

Our experiment demonstrates the use of the Rules as Code methodology to detect a drafting issue in a proposed statutory text, and to verify the effect of a proposed amendment. The issue discovered in this experiment is the type of issue that Rules as Code is intended to address early: one that if left unaddressed negatively affects the degree to which the statutory text can be automated.

With regard to s(CASP)'s strengths and weaknesses for this task, the access to "why not" queries and natural language justifications was extremely valuable both in the encoding of the Rule, and in the analysis of test failures. s(CASP)'s abductive reasoning over constraints, and the fact that it returned answer sets rather than bindings, allowed the author to test the encoding against a wide variety of fact scenarios simultaneously, quickly providing a deep level of insight into the behaviour of the encoding. s(CASP) also facilitated the use of a version of defeasibility that allowed defeating relations of both the "subject to" and "despite" types to be encoded where they appear in the text, enhancing maintainability of the code [7].

s(CASP)'s abductive queries slow down considerably with the complexity of the code, and so it may not be an appropriate approach for real-time applications of abductive reasoning. However, its performance on deductive reasoning tasks was very efficient, completing the 25 tests in this experiment in an average of less than 1 second each, which suggests it can also be used to answer legal questions with complicated fact scenarios and complicated rules in a user-facing application.

ACKNOWLEDGMENTS

I owe a debt of gratitude to all my colleagues at the SMU Centre for Computational Law, and in particular our Principal Investigator Meng Weng Wong, Industry Director Alexis Chun, and Professors Lim How Khang and Jerrold Soh, all of whom contributed greatly to the legal analysis. Professors Gopal Gupta of University of Texas at Dallas and Joaquín Arias at Universidad Rey Juan Carlos provided valuable assistance on the effective use of s(CASP). The feedback of the reviewers has also improved the paper and is gratefully acknowledged.

This research is supported by the National Research Foundation (NRF), Singapore, under its Industry Alignment Fund – Pre-Positioning Programme, as the Research Programme in Computational Law. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] L. Allen and C. R. Engholm. 1978. Normalized Legal Drafting and the Query Method. *Journal of Legal Education* 29 (1978), 380–412.
[2] Joaquín Arias, Manuel Carro, Zhuo Chen, and Gopal Gupta. 2020. Justifications for Goal-Directed Constraint Answer Set Programming. *arXiv preprint arXiv:2009.10238* (2020).
[3] Joaquin Arias, Manuel Carro, Elmer Salazar, Kyle Marple, and Gopal Gupta. 2018. Constraint answer set programming without grounding. *Theory and Practice of Logic Programming* 18, 3-4 (2018), 337–354.
[4] Organization for Economic Cooperation and Development Observatory for Public Sector Innovation. [n.d.]. Cracking the Code: Rulemaking for humans and machines. Accessed February 28, 2021, at https://oecd-opsi.org/wp-content/uploads/2020/10/Rules-as-Code_Highlights_Final_HighRes.pdf.
[5] D. Merritt. 2017. *Expert Systems in Prolog*. Independently Published. https://books.google.com.sg/books?id=6IQGyQEACAAJ
[6] Marek J. Sergot, Fariba Sadri, Robert A. Kowalski, Frank Kriwaczek, Peter Hammond, and H. Terese Cory. 1986. The British Nationality Act as a logic program. *Commun. ACM* 29, 5 (1986), 370–386.
[7] Hui Wan, Benjamin Grosof, Michael Kifer, Paul Fodor, and Senlin Liang. 2009. Logic Programming with Defaults and Argumentation Theories. In *Logic Programming*, Patricia M. Hill and David S. Warren (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 432–448.