Written Proposal

Dylan Jenkins

February, 26, 2022

1 Introduction & Problem

Computer Science has many applications across many fields. However, one field wherein computer science's influence is not often observed in is the field of law. Aside from sifting through emails and documents to sort between useful and not, lawyers do not often use computers in meaningful ways in their profession. However, I believe there exists a vital part of the lawyers job that may be susceptible to more influence. Logic is a cornerstone in computing to make decisions. Likewise, logic is used in law to find contradictions between facts and laws themselves. I believe it is possible to use computers to find contradictions given these laws.

I pose the following question: Can a computer be used to codify criminal laws and find contradiction within the laws themselves?

2 The Literature & Interest

There are a few projects that have been attempted to codify laws into computers. A Korean development team was able to codify building permit laws into a computer to determine who should receive building permits. This project translated the laws themselves into their component parts, and used a programs to analyze these parts in order to use the law when determine when to give a building permit [2]. Another project regarding engineering laws was able to codify contradictions into computers to help solve their problems. It has a very particular way of defining contradictions which I believe is useful [1]. Lastly, there was an article which talked about a programming language which handles tax law. This article mainly focuses on the language itself, but it does provide insight on to how laws were defined using that language [3].

From the above literature, it is evident that there exists interest in the field of converting laws into computer executable code. However, throughout my research I could not find a program which took federal laws, codified them, and

used them in some executable form. So there is a problem. With the interest in the subject of law displayed by those articles as well as my own, I believe this meets the requirement of an unsolved problem which has interest in the greater field.

3 Methodology

The article mentioned previously about contradictions in engineering laws provided a decent definition of what constituted a contradiction. It provides a framework for defining a contradiction using Elements (Objects), Parameters (Description of Elements), and Values (Properties of Parameters) [1]. Using this methodology, I seek to make a proof of concept using this framework alongside college courses to find contradictions in student's schedules as well as contradictions between courses themselves and majors as a whole. In this regard, majors will be a parallel to crimes and courses would be a parallel to facts required for said crimes. The students in this model will be parallels with suspects. Students have courses they have taken which informs whether they can take a given course and how much progress they have made towards a major. I believe this should make for a good model, and it may help the college find some contradictions with their current course roster and major requirements. To codify both majors and courses, I will be using the methodology from the building permit article. Namely, I will be simplifying courses and majors to their base elements and codifying them this way. These base elements I have found are: Course ID, Hard Requirements (Courses that must be taken prior), and Soft Requirements (x number of courses must be taken from a group of courses with size $y > x$). I will be using prolog to complete this project as it would make the most sense when replacing our factors with that of law.

4 Progress

So far, I have codified some courses and majors and have the ability to determine whether a student can take a course or has taken the requisite courses for a section of a major or the entire major. I have also coded a few contradictions which should help to find any problems with courses listed on the school's website. I have codified about one-third of the courses listed on the website.

I would like to codify as many majors as possible to run through the contradictions I am coding. After I run the courses through these contradictions, I will record the contradictions I find and put them into my final report.

5 Time Table

February 28th - March 4th - Finish Codifying courses and begin codifying majors.

March 14th - March 19th - Finish Codification of majors. Implement any further contradictions thought of during the previous weeks.

March 21st - April 2nd - Test contradictions on majors and courses and record them for the final report. Write final report. Create presentation based on final report.

April 4th - April 9th - Proof-read final report draft, prepare code for final submission.

April 11th - April 16th - Refine final report based on feedback, Work on presentation.

April 18th - April 23rd - Finish oral presentation, practice oral presentation.

References

[1] Denis Cavallucci, Francois Rousselot, and Cecilia Zanni-Merk. On contradiction clouds. *Procedia Engineering*, 9:368–378, 12 2011.

[2] Hyunsoo Lee, Jin-Kook Lee, Seokyung Park, and Inhan Kim. Translating building legislation into a computer-executable format for evaluating building permit requirements. *Automation in Construction*, 71:49–61, 2016. The Special Issue of 32nd International Symposium on Automation and Robotics in Construction.

[3] Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko. Catala: a programming language for the law. *Proceedings of the ACM on Programming Languages*, 5(ICFP):1–29, Aug 2021.