1 Logic Programming for Large Scale Applications in Law: A Formalisation of Supplementary Benefit Legislation, Trevor Bench-Capon, George Robinson, Tom Routen, and Marek Sergot (Adam Wyner)

The 1980s saw the rise of one of central projects of AI and Law - the representation of law as executable logic programs. [1] describes one of the steps in the development of the project with the representation of a large portion of the United Kingdom's legislation on "Supplementary Benefit" (SB) as an executable logical model of law in Prolog. It outlines the legislation, the task the representation supports, and problems in creating the representation.

One of the earlier, smaller stages in the project were [6], which represented the "British Nationality Act 1981" (BNA) in Prolog. More broadly, it was part of the efforts in other fields to develop expert systems and executable representations of knowledge in general and legal knowledge in particular [7, 5, 3]. The BNA study was a rather constrained, focused, feasibility study of a self-contained article of law that led to a reasonably sized application. There was no method and solutions were ad hoc.

The study with SB was intended to see what would happen with a large piece of legislation that had been revised and that interacted with other laws. It was funded by the government of the United Kingdom to create Artificial Intelligence software that supported the application of legislation by government offices, in this case, the Department of Health and Social Services (DHSS), which was a large, complex, organisation tasked with applying social legislation. The SB itself is an extensive piece of legislation (600 pages) including definitions; asides from the legislation, there were auxiliary documents such as a guidance manual. In the end, 90 percent of the legislation was coded over a period of two months, though unsupervised and not evaulated.

They distinguish requirements for users of the legislation, e.g. clerks who apply the law need different information, solicitors who advise on how the law is likely to be applied, and individuals or organisations that want to comply with the law. Rather than creating different representations of the same legislation, which would lead not only to redundancy, but possible inconsistency and unclarity, it was proposed to create a logical, executable formalisation of the legislation that could be used as a core across requirements and upon which additional functionalities could be built. For example, it may be useful to provide determinations (from facts to inferences), explanations (from inferences to their proofs), or as compliance monitors (checking that procedures are adhered to). Thus, to maintain a link between the source legislation and the code, the legislation itself is the basis of the translation to logic programming. This was in contrast to [3] that ignored the legislation itself and created an expert system, such as how a clerk might apply the legislation.

The approach was to code directly, starting top-down, based on a reading of the legislation, and without a specific methodology or discipline. High-level concepts were defined in terms of lower level concepts, and eventually grounded in facts in a database or supplied by the user. This rule-based approach contrasts with [7], which has a preliminary analysis of legislation in terms of entities

and relationships. Nor was there an intermediate representation, e.g. a 'structured english' form of the legislation, which would have clarified the analysis and supported verification. Thus, complex predicates were created that could be decomposed as in:

X is-not-required-to-be-available-for-employment *if*
X is-regularly-and-substantially-engaged-in-caring-for-a-severely-disabled-person.

X is-regularly-and-substantially-engaged-in-caring-for-a-severely-disabled-person *if*
X is-regularly-and-substantially-engaged-in-caring-for **Y**,
Y is-severely-disabled.

There were, then, substantial issues about what predicates the formalisation should include and what should be decomposed. The guiding principle was to decompose as little as possible.

In the course of the analysis, some general issues were identified, two which we mention. First, a basic definition of an entitlement may have enabling provisions that specify exceptions, some of which are in statutory instruments such as regulations. Not only did such provisions have to be defined, e.g. as negations of predicates in the body of a rule, but the relation between the Act and other statutory instruments had to be coordinated, which was problematic and raised the issue of what was being represented. In practice, reference to clauses were incorporated into complex predicates, though there could be long chains of such references. They proposed alternatively to create a database of the structure of the legislation, which would be maintained and used for reference. Another general issue was representing and reasoning with temporal information as objects and relationships change in time through the influence of events that occur in the world. For example, an individual's entitlement may change over time. In addition, there are ideosyncratic definitions in the legislation for temporal periods, e.g. what counts as a continuous temporal period may include breaks, or retrospective rules, e.g. as in receiving a benefit as the result of appealing a decision. For such issues, the suggestion was to introduce some knowledge management system and the means to reason with time. Other issues were the treatment of "deeming provisions", negation, extension to case law, and open texture concepts.

This line of research, where legislation is translated into executable logic programs, has been a notable commercial success. The company Softlaw adopted key ideas and evolved to provide large scale, web-based tools to serve legislation to the public [4, 2]. In addition to executable logic, Softlaw scoped the problems, maintained links to the orginal legislative source, added a controlled language, among other features. The company has been (after several intervening changes) acquired by Oracle, where it now provides Oracle Policy Management to governments the world over. Despite this development and success, the AI and Law research community seems not to have followed suit with similar open-source tools for research and development.

References

[1] T. Bench-Capon, G. Robinson, T. Routen, and M. Sergot. Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In *International Conference on Artificial Intelligence and Law*, pages 190–198, 1987.

[2] S. Dayal and P. Johnson. A web-based revolution in australian public administration. *Journal of Information, Law, and Technology*, 1, 2000. Online.

[3] P. Hammond. *Representation of DHSS Regulations as a Logic Program.* Department of Computing: Research report DoC. 1983.

[4] P. Johnson and D. Mead. Legislative knowledge base systems for public administration: Some practical issues. In *ICAIL*, pages 108–117, 1991.

[5] P. Leith. ELI: An expert legislative consultant. In *Proceedings of the IEE Conference on Man/Machine Systems UMIST*, number 212. Conference Publication, 1982.

[6] M. Sergot, F. Sadri, R. Kowalski, F. Kriwaczek, P. Hammond, and T. Cory. The British Nationality Act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986.

[7] R. Stamper. LEGOL: Modelling legal rules by computer. *Computer Science and Law*, pages 45–71, 1980.