On the Modelling and Analysis of Regulations

G. Antoniou, D. Billington, G. Governatori and M.J. Maher
CIT, Griffith University
Nathan, QLD 4111, Australia
{ga,db,guido,mjm}@cit.gu.edu.au

Abstract

Regulations are a wide-spread and important part of government and business. They codify how products must be made and processes should be performed. Such regulations can be difficult to understand and apply. In an environment of growing complexity of, and change in, regulation, automated support for reasoning with regulations is becoming increasingly necessary. In this paper we report on ongoing work which aims at providing automated support for the drafting and use of regulations using logic modelling techniques. We highlight the support that can be provided by logic modelling, describe the technical foundation of our project, and report on the status of the project and the next steps.

Keywords

1 Introduction

Regulations are a wide-spread and important part of government and business. They codify how products must be made and processes should be performed. Such regulations can be difficult to understand and apply. Even stand-alone regulations can be self-contradictory, as a result of the incremental process of their development and the lack of a formal drafting process. The problem becomes more difficult when independently developed regulations apply to a situation. For example, when two regulations overlap, it is not clear whether one regulation takes precedence or both regulations apply. Even when regulations are formally drafted, as is often done in the legal domain, problems with the consistency, interpretation and use of regulations still remain (e.g. Sergot et al. (1986)). In an environment of growing complexity of, and change in, regulations, automated support for reasoning with regulations is becoming increasingly necessary.

Previous attempts using logic modelling includes, among others: Prakken and Sartor's (1996) analysis of legal argumentation; Gordon's (1995) The Pleadings game, implementing Article Nine of the Uniform Commercial Code of the United States (UCC), and intended to support decisions about issues raised during the pleading phase of legal proceedings; Gardner's (1987) model of legal reasoning; and McCarthy's (1995) TAXMAN II system.

Lessons learnt from previous attempts include the following: (i) the underlying modelling technique must be supported by sufficiently efficient reasoning tools; (ii) key phenomena of the problem must be representable in the chosen formalism; (iii) *naturalness of expression* (the ability to represent the problem at hand in a transparent and natural way) is a key issue

to success; and (iv) logic is not sufficient on its own – pragmatic aspects must be taken into account.

This paper reports on a major Australian project which applies logic modelling to regulation analysis. In the rest of the paper we will discuss the benefits of logic modelling in the domain of regulations, present the basic notions of the modelling formalism used in our project, and will describe the current status of the project.

2 How Logic Modelling Supports the Analysis of Regulations

The use of logic modelling techniques is beneficial for regulatory reasoning in various ways. In the following we distinguish between drafting regulations, and understanding and applying regulations.

Regarding the *understanding and application of regulations*, formal systems have the following advantages. These advantages are important, for example, for "naive users/subjects of regulation" who are regulated but do not wish to study the regulations.

1. *Decision support:* It is possible to run a specific case with the given regulations to get a correct answer. Such a case might be a query whether a certain damage is covered by an insurance policy.

2. *Explanation:* When an answer is given, there is also a reasoning chain explaining this response. This can be most useful in, say, help desks.

Drafting regulations can be supported in the following ways:

3. *Anomaly detection:* Formal methods can be used to detect anomalies such as inconsistency, incompleteness and circularity. Such anomalies are detected either by static analysis, or by the performance of the proof theory.

4. *Hypothetical reasoning:* It is possible to investigate the effects of changes to regulations on the entire regulatory system. This is possible because regulations are represented as executable specifications. For example, the Taxation Office is interested in detecting possible loopholes.

5. *Debugging:* In many cases we know what the answer to a specific query should be, yet the regulations in their current form lead to a different answer. Debugging suggests changes to the regulations which will have as an effect the desired outcome. In our project, debugging can be carried out along the lines of "declarative debugging" (Naish, 1997).

3 The Logical Foundation: Defeasible Logic

In our project we use *defeasible rules* with *priorities* as a logical method of analysing regulations. Rules are normally sufficiently expressive to represent single items within a regulation. Since regulations may contradict one another the use of *defeasible* rules is indicated: these are rules that do not necessarily fire; instead they may be blocked by other rules with contrary conclusions.

Regulations commonly include exceptions. For example, in the Australian Civil Aviation Regulations 1988, rule 162 (1) states: "When two aircraft are on converging headings at approximately the same height, the aircraft that has the other on its right shall give way, except that (a) power-driven heavier-than-air aircraft shall give way to airships, gliders and balloons; ..." (Aneiros, 1999). In addition there may be principles by which one set of regulations is superior to another (for example, a regulation of a higher authority is superior to a regulation of a lower authority; or a more recent regulation might override an older regulation). The priorities on defeasible rules are an adequate and natural way of representing these information.

For these reasons we believe that defeasible rules and priorities offer a *natural* way of representing regulations. The necessary reasoning support will be provided by a particular logic we will be using: defeasible logic (Nute, 1987; Billington, 1993). It is an approach to sceptical nonmonotonic reasoning that has a very distinctive feature: It was designed to be easily implementable right from the beginning, and has low computational complexity.

3.1 Basics of Defeasible Logic

Defeasible logic is a sceptical formalism, meaning that it does not support contradictory conclusions. Instead it seeks to resolve differences. In cases where there is some support for concluding A but also support for concluding $\texttt{not}\ A$, the logic does not conclude either of them (thus the name "sceptical"). If the support for A has priority over the support for $\texttt{not}\ A$ then A would be concluded. Sceptical reasoning is appropriate for the study of regulations. Users of regulations are mostly interested in getting correct advice without being confronted with conflicting views. Drafters of regulations can detect an anomaly of the regulations from a conflict that cannot be resolved.

A set of regulations will be represented as a defeasible theory. A defeasible theory, i.e., a knowledge base in Defeasible Logic, consists of four different kinds of knowledge: facts, strict rules, defeasible rules, and a superiority relation.

Facts denote simple pieces of information that are deemed to be true regardless of other knowledge items. A typical fact is that John is a minor: $\texttt{minor}(\texttt{John})$.

Briefly, strict and defeasible rules are represented, respectively, by expressions of the form $A_1, \ldots, A_n \rightarrow B$ and $A_1, \ldots, A_n \Rightarrow B$, where $A_1, \ldots, A_n$ is a possibly empty set of prerequisites and B is the conclusion of the rule.

Strict rules are rules in the classical sense: whenever the premises of a rule are given, we are allowed to apply the rule and get a conclusion. When the premises are indisputable (e.g. facts) then so is the conclusion. An example of a strict rule is "every minor is a person". Written formally:

$$\texttt{minor}(X) \rightarrow \texttt{person}(X).$$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is "every person has the capacity to perform legal acts to the extent that the law does not provide otherwise"; written formally:

$$\texttt{person}(X) \Rightarrow \texttt{hasLegalCapacity}(X).$$

The idea is that if we know that someone is a person, then we may conclude that he/she has legal capacity, *unless there is other evidence suggesting that he/she has not*.

The *superiority relation* among rules is used to define priorities among rules, that is, where one rule may override the conclusion of another rule. For example, given the defeasible rules

$$\begin{array}{ll} r: & \texttt{person}(X) \Rightarrow \texttt{hasLegalCapacity}(X) \\ r': & \texttt{minor}(X) \Rightarrow \texttt{not hasLegalCapacity}(X) \end{array}$$

which contradict one another, no conclusive decision can be made about whether a minor has legal capacity. But if we introduce a superiority relation $>$ with $r' > r$, then we can indeed conclude that the minor has not legal capacity.

It turns out that we only need to define the superiority relation over rules with contradictory conclusions. Also notice that a cycle in the superiority relation is counterintuitive from the knowledge representation perspective. In the above example, it makes no sense to have both $r > r'$ and $r' > r$. Consequently, the defeasible logic we discuss requires an acyclic superiority relation.

A conclusion B is supported if there is a rule whose conclusion is B, the prerequisites are either supported or given in the case at hand, and a stronger rule whose conclusion is not B has prerequisites that fail to be supported.

To explain the mechanism of defeasible derivations we consider again rule 162 of the Australian Civil Aviation Regulations 1988; it can be represented in defeasible logic as follows:

$$r_1 : \texttt{onTheRightOf}(\texttt{X}, \texttt{Y}) \Rightarrow \texttt{rightOfWay}(\texttt{X}, \texttt{Y})$$

stating that the aircraft X has right of way over the aircraft Y if X is on the right of the Y;

$$r_2 : \texttt{powerDriven}(\texttt{X}), \texttt{not powerDriven}(\texttt{Y}) \Rightarrow \texttt{not rightOfWay}(\texttt{X}, \texttt{Y})$$

saying that a power-driven aircraft has not right of way over a non-power-driven one.

$$\begin{array}{c} r_3 : \texttt{balloon}(\texttt{X}) \rightarrow \texttt{not powerDriven}(\texttt{X}) \\ r_4 : \texttt{glider}(\texttt{X}) \rightarrow \texttt{not powerDriven}(\texttt{X}) \end{array}$$

r_3 and r_4 classify balloons and gliders as non-power-driven aircraft and

$$r_5 : \Rightarrow \texttt{powerDriven}(\texttt{X})$$

assumes aircraft to be power-driven unless further information is given. The superiority relation is determined as follows: $r_2 > r_1$ because r_2 is an exception to r_1 and, by specificity

$$r_3 > r_5 \qquad r_4 > r_5$$

Let us examine the following cases: 1) two aircraft of the same type (power-driven, non-power-driven) are converging 2) a power-driven aircraft and a non-power-driven aircraft are converging. In the first case we can apply r_1 since the prerequisites of r_2 do not hold. In the second case we can apply r_1 given that both the prerequisites obtain.

3.2 Examples: University regulations

Example 1: Academic misconduct
A typical rule taken from the Griffith University policy on academic misconduct:

> Where a student has been found guilty of academic misconduct on more than one occasion and has previously been penalised as set out in 3.1–3.3 above, the penalty shall be normally the exclusion from the course, unless in the opinion of the relevant Assessment Board there are mitigating circumstances.

This is a typical *rule with exceptions*. In the framework we are proposing, we would represent this rule as follows:

r_1 : $\texttt{guilty}, \texttt{repeat}, \texttt{prevPenalised} \Rightarrow \texttt{exclude}$
r_2 : $\texttt{mitigatingCircumstances} \Rightarrow \texttt{not exclude}$
$r_2 > r_1$

Notice that the predicate `mitigatingCircumstances` will be only estabilished if the Assessment Board decides so. Then a fact would be added to this particular case, and the decision would be not to exclude. This example illustrates the representation of exceptions in defeasible logic: both general rules and their exceptions are formalised as defeasible rules. An exception is stronger than a general rule.

Example 2: Guidelines on fees
The second example is more substantial. It comprises part of the Griffith University guidelines on fees.

1.1 The University may not charge tuition fees for Australian students in undergraduate award courses.

1.2 The University may charge fees for postgraduate courses.

1.3 Overseas students are generally fee paying but there are some exceptions. There is minimum fee level set by the Government for fees for overseas students (FPOS). There are special arrangements for international exchange students. Refer to the GU International Center for advice on these issues.

1.7 All students are liable for HECS (Higher Education Contribution Scheme) with very few exceptions. Students who do not pay HECS include:

- FPOS students
- fee paying postgraduate students
- non-award students
- students with an APA (Australian postgraduate award)
- students wholly sponsored by an employer.

Here is the representation of this information in defeasible logic:

$r_{1.1}$: $\texttt{student}, \texttt{australian}, \texttt{undergrad} \Rightarrow \texttt{not fee}$

$r_{1.2}$: $\texttt{student}, \texttt{postgrad} \Rightarrow \texttt{fee}$

$r_{1.3a}$: $\texttt{student}, \texttt{overseas} \Rightarrow \texttt{fee}$

$r_{1.3b}$: $\texttt{student}, \texttt{overseas} \Rightarrow \texttt{payFPOS}$

$r_{1.3c}$: student, overseas, exchange $\Rightarrow$ not payFPOS

$r_{1.7a}$: student $\Rightarrow$ HECS

$r_{1.7b}$: student, payFPOS $\Rightarrow$ not HECS

$r_{1.7c}$: student, postgrad, fee $\Rightarrow$ not HECS

$r_{1.7d}$: student, nonAward $\Rightarrow$ not HECS

$r_{1.7e}$: student, APA $\Rightarrow$ not HECS

$r_{1.7f}$: student, fullySponsored $\Rightarrow$ not HECS

$r_{1.3c} > r_{1.3b}$

$r_{1.7b} > r_{1.7a}$

$r_{1.7c} > r_{1.7a}$

$r_{1.7d} > r_{1.7a}$

$r_{1.7e} > r_{1.7a}$

$r_{1.7f} > r_{1.7a}$

Example 3: Exam timetabling policy

1.0 For subjects in which a final examination accounts for a major portion of the student's assessment, a one-week revision period should be established between the end of formal content presentation and the commencement of examinations.

2.0 Week 15 should be established as the examination revision week, with the examination period extending from the Saturday of Week 15 through Weeks 16 and 17 as required.

3.0 Courses which assess predominantly by continuous assessment will not be required to comply with the semester structure of 2.0. For these courses, the teaching period may extend past Week 14. These courses will be identified via the existing course approval process.

4.0 Where a Faculty wishes to ensure a satisfactory examination schedule by scheduling a course's examination towards the end of Week 15, this may be done, provided that formal content presentation ceases one week before the first exam.

It is clear that again we are faced with the phenomenon of rules with exceptions, which may be expressed using defeasible rules and priorities.

This example highlights one sort of anomaly likely to be uncovered by a formal analysis of regulations, namely redundancy. For subjects in which a final examination accounts for a major portion of assessment, 4.0 is subsumed by 1.0. Also 4.0 essentially extends 1.0 to new cases (subjects where a final exam is a minor assessment item). Thus items 1.0 and 4.0 can (and should) be reasonably combined to one item.

We will refer to this example later on when we raise the question of which features are needed for the analysis of regulations, beyond the features already available in defeasible logic.

Remark: Explicit priority information

As we saw in section 3 defeasible logic makes use of an explicit superiority relation. This may seem questionable, since it places the burden on the user to provide the priority information. Our response to this criticism is as follows:

1. Indeed it is possible in defeasible logic to extract superiority relations from a set of rules based on specificity (see Billington, 1990; Nute, 1994).

2. Implicit priority criteria such as specificity are not always capable of capturing *all* the prioritisation information.

3. Regulations often include policies of prioritisation, which then is naturally represented in an explicit way. The following example illustrates this point.

Example 4: Credit transfer policy

> 1.1 . . . The policy applies to all coursework award courses of the University; however, the award of credit in honours and masters courses will be restricted by specific policy applying to these courses (refer to *Requirements and Administration of Honours Courses*, and *Masters Degree by Coursework Rules*).

From this regulation it is clear that while the rules in the *Credit Transfer Policy* may apply to postgraduate courses, they may be overruled by those in the other documents referred to, which should thus be given higher priority.

4 Report on the Project

4.1 Current State

An implementation of the underlying formalism, defeasible logic, is operational. We have also embarked on substantial case studies regarding the modelling of regulations in our chosen formal language, and the requirements that go beyond the logical formalisms (see next subsection). The domains we are investigating include health regulations, aviation regulations, council laws and university regulations (Aneiros, 1999). Currently we are exploring the possibilities of a collaboration with the Australian Taxation Office on the detection of loopholes in taxation laws and regulations.

The next steps in our project include the following: (i) the determination and implementation of extra-logical features (see next subsection); (ii) the implementation of a user-friendly system which supports the practical modelling, analysis and maintenance of regulations (see subsection 4.3); and (iii) the completion of the case studies and their evaluation (see subsection 4.4).

4.2 Beyond Defeasible Logic

In the following we discuss briefly some features that appear to be useful or necessary for the analysis of regulations, and which go beyond defeasible logic. We believe that some of the earlier attempts to model and reason with regulations failed because such aspects were neglected (mainly due to their perceived lack of theoretical interest). Our project aims at solving the problem instead of demonstrating the superiority of a modelling method over other methods.

Hierarchies of regulations: Often regulations themselves are organised in a hierarchical fashion. For example on top of university regulations there exist public service regulations which, if a conflict should arise, are stronger than university regulations. The idea of using rules in an inheritance network system has recently been propagated by researchers at the IBM T.J.

Watson Research Center (Morgensten and Singh, 1997). Moreover Prakken and Sartor (1996) and Gordon (1995) propose to represent hierarchical principles such as *lex posteriori* (i.e., more recent rules have higher priority than older ones) and *lex superiori* (i.e., rules originated from higher sources have precedence over rules coming from lower sources) using a priority relation and defeasible rules.

Arithmetic and temporal capabilities: Sometimes simple arithmetic and temporal operations are required. For example, in regulations which contain time conditions we need elementary date operations (for example, week 14 comes immediately after week 13). This can be achieved by embedding in the system simple mathematical tools and calendar logics.

Ontological knowledge: Typically regulations are not given in an empty environment; instead they make use of terminology and concepts which are relevant to the organisation and/or the aspect they seek to regulate. Thus, to be able to capture the meaning of regulations, one needs to encode not only the regulations themselves, but also the underlying ontological knowledge. This knowledge usually includes the terminology used, its basic structure, and integrity constraints that need to be satisfied.

An example of terminological problems is found in the Griffith University examination timetabling policy, where the old terms "course" and "degree" are sometimes used instead of the current ones "subject" and "course" (!!) respectively.

4.3 Features of the Implementations

As we have already said the implementation in Prolog of Defeasible Logic is already operational (cf. Covington et al. (1997); Maher and Governatori (1999)). On the other hand we have planned to develop a substantially complete prototype application with two main modules: the first with particular emphasis on drafting and analysis of regulations and the second on decision support. The decision support module is meant primarily to be used by customer support services while the other module is intended as a tool for the development and analysis of regulations.

Although the two parts have different purposes we have identified some common functionalities for the user-interfaces. In particular, both should be able to load regulations on demand and be responsive to queries. Furthermore, they have to highlight the rules relevant for the cases at hand; moreover, the drafting module has also to reproduce the steps that lead to the conclusions; this would be particularly useful for the analysis and debugging of regulation drafts. In this way it will be able to detect and report anomalies and, in such cases, it could suggest possible changes to the rules.

The decision support module should be equipped with a query editor (see, for an example of a query editor applied to regulations, Gantner (1998)), i.e., a series of templates that guide the user to formulate queries suitable to be processed successfully.

A very important part of the drafting module is the rule/regulation editor. Such a tool should provide templates to produce at the same time a version of a regulation suitable to be understood by humans, and another version to be processed by the system itself. In this respect we intend to follow the model proposed by Norma-Editor, developed at the University of Bologna for Bologna City Council, based on SGML and where rules are defined by appropriate DTDs (see, Ballerini et al. (1998)).

4.4 Evaluation

Due to the nature of regulations and the intended applications we believe that a quantitative evaluation is not appropriate. So we plan a twofold qualitative evaluation. On one hand, we will adopt standard techniques to measure user satisfaction (see, for a survey, DeLone and McLean (1992)). In particular, as far as the drafting module is concerned we shall carry out interviews on a study group of lawyers and legal drafting experts to test the ability of the system to detect previously unnoticed contradictions, loopholes and ambiguity (see the end of section 3). The decision support module will be evaluated for its user-friendliness, efficiency and accuracy based on a group study composed by customer support staff members.

On the other hand, due to the dynamic nature of regulations, we have also to evaluate internally how easy the system is to maintain and implement, and how simple the migration from a prototype developed using the drafting module to the corresponding decision support system is.

5 Conclusion

Regulations play an important role in the organisation and functioning of society in general, and business in particular. The increasing complexity of regulations and the frequency of necessary changes, mainly due to technological change and the current trend towards globalisation, make automated support in the analysis of regulations necessary. In this paper we report on our ongoing work on the application of logic modelling, in particular defeasible reasoning, to the analysis, drafting and use of regulations.

So far we have established and implemented the logical foundations of the framework and we have carried out initial case studies (modelling of regulations in several domains). The preliminary results seem very promising, showing a close correspondence between regulations and their formalization.

References

M. Aneiros. *Logical Methods for Modelling Regulations – Report*. Griffith University research report, 1999.

J.P. Ballerini, A. Capelli, M. Palmirani, G. Sartor and F. Vitali. Norma-Editor: An Integrated Editor for Handling and Consolidating Legal Texts. In: A. Artosi, M. Atienza and H. Yoshino (eds), Legal Computer Science. Clueb, Bologna, 1998.

D. Billington, K. de Coster and D. Nute. A modular translation from defeasible nets to defeasible logic. *Journal of Experimental and Theoretical Artificial Intelligence* 2 (1990): 151-177.

D. Billington. Defeasible Logic is Stable. *Journal of Logic and Computation* 3 (1993): 370–400.

M.A. Covington, D. Nute and A. Vellino. *Prolog Programming in Depth*. Prentice Hall 1997.

W.H. DeLone and E.R. McLean. Information Systems Success: The Quest for the Dependent Variable. *Information System Research* 3 (1992): 60–95.

A. Gardner. *An Artificial Intelligence Approach to Legal Reasoning*. MIT Press 1987.

F. Gantner. Forms and Legal Informatics. In A. Artosi, M. Atienza and H. Yoshino (eds.): *Legal Computer Science*, Clueb, Bologna, 1998.

T. Gordon. *The Pleadings Game*. Kluwer 1995.

M.J. Maher and G. Governatori. Semantic Decomposition of Defeasible Logics. In *Proc. American National Conference on Artificial Intelligence (AAAI-99)*, to appear.

L.T. McCarty. An Implementation of Eisner vs. Macomber. *Proc. 2nd International Conference on Artificial Intelligence and Law*, ACM Press 1995, 276–268.

L. Morgenstern, M. Singh. An Expert System Using Nonmonotonic Techniques for Benefits Inquiry in the Insurance Industry. In *Proc. IJCAI-97*, Morgan Kaufmann 1997.

L. Naish. A Declarative Debugging Scheme. *Journal of Functional and Logic Programming* 3 (1997).

D. Nute. Defeasible Reasoning. In *Proc. 20th Hawaii International Conference on Systems Science*, IEEE Press 1987, 470–477.

D. Nute. Defeasible Logic. In D.M. Gabbay, C.J. Hogger and J.A. Robinson (eds.): *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3*, Oxford University Press 1994, 353-395.

H. Prakken and G. Sartor. A Dialectical Model of Assessing Conflicting Arguments in Legal Reasoning. *Artificial Intelligence and Law* 4 (1996): 331–368.

M.J. Sergot, F. Sadri, R.A. Kowalski, F. Kriwaczek, P. Hammond and H.T. Cory. The British Nationality Act as a Logic Program. *Communications of the ACM* 29,5, 370-386.

Acknowledgments

This research was supported by the Australia Research Council under Large Grant No. A49803544.

Copyright