Logic Programming for Large Scale Applications in Law: A Formalisation of Supplementary Benefit Legislation

T.J.M.Bench-Capon, G.O.Robinson, T.W.Routen, M.J.Sergot

Department of Computing
Imperial College of Science and Technology
180 Queen's Gate
London SW7 2BZ, England

INTRODUCTION

The paper describes a project which began in April 1986 on the representation in logic of a large portion of the United Kingdom's welfare law; specifically the legislation relating to 'Supplementary Benefit'. The paper describes the nature of the legislation and the kinds of tasks which the ıepresentation is intended to support, reports on the progress of the project so far, and describes some of the knowledge representation problems that have arisen and how we propose to overcome them. The principal aim of the paper is to elaborate on our use of logic programming techniques for applications in law, and on our view of how executable logical models of law can be used to support various aspects of legal reasoning.

Within the Logic Programming Group at Imperial College the domain of law and legal reasoning is seen as a primary source of applications. These applications are intended both to test out developing techniques of logic programming, and to suggest and motivate future extensions. They range in scope from simple systems which can apply a fixed set of legal rules, to more ambitious systems which address other aspects of the legal problem solving process.

In general terms, our approach is to represent some appropriate fragment of legislation as a logical model, and then to execute the model in ways in which it can support legislation related applications. Several small scale projects have been undertaken to establish the fitness of this approach. The most widely known of these is the project relating to the British Nationality Act 1981, described in [12]. A more general account of the approach, together with references to other smaller formalisations, is provided in [11]. The results of this work have been encouraging and demonstrated that the approach is worth pursuing.

We see the law as an ideal domain within which to try out proposed techniques of knowledge representation. We are exploring such questions as the representation of time (see later), and general proposals for embedding the use of logic within a flexible framework that can adjust as new knowledge has to be assimilated (see for example [7, Chapter 13]). Such systems can be built in principle, but further investigation was (and is) required before the techniques can be applied in practical applications. We are currently investigating the extent to which these general ideas can be applied to reasoning with case law and precedent in the resolution of open texture [1].

The formalisation of the British Nationality Act drew attention to a number of different and complementary problems, however. The British Nationality Act project was undertaken to demonstrate the computational feasibility of applying existing logic programming techniques to build a realistically sized application in law. We placed less emphasis on knowledge representation and concentrated on producing a usable system that covered as much of the Act as possible. We tended to adopt short term, relatively ad hoc solutions to problems of knowledge representation, rather than divert to detailed and more general investigation of the problems we encountered. In this respect the British Nationality Act project and subsequent experiments were reasonably successful. They supported our belief that a large system could be built fairly straightforwardly using standard logic programming techniques (as long as we were prepared to adopt ad hoc solutions to difficult problems of knowledge representation). In other respects, however, it was harder to formalise the British Nationality Act than we expected because problems arose mainly from the scale of the task. Such problems could be expected to multiply if a larger piece of legislation were tackled.

The British Nationality Act and the other fragments of legislation which were attempted subsequently are representative of the kinds of legislation which would form the basis for many practical applications. Nevertheless, they do incorporate important simplifications. They are, on the whole, relatively small, clearly written and self-contained.They might therefore be considered atypically amenable to the approach we adopted. Partly in order to meet this kind of objection, we felt that there was a need to embark on the formalisation of a larger and more representative piece of legislation. More importantly, we wanted more experience with the process of constructing a substantial formalisation.

Supplementary Benefit

The law relating to Supplementary Benefit is a large piece of legislation by any standards. Apart from its size, it has many other complicating factors. For example, it has undergone considerable revision and amendment since its introduction, and it interacts with many other laws, including education law and employment law. There was, however, a second reason why we should choose this particular fragment of law as the subject matter for the project.

The Alvey Directorate of the United Kingdom's Department of Trade and Industry is currently funding a large programme on various aspects of information technology. As part of this programme, a number of "Large Demonstrator Projects" have been established whose aim is to investigate and demonstrate the use of Artificial Intelligence techniques in large practical applications. One such Demonstrator is investigating the provision of decision support systems to large legislation based organisations. This project is taking as its focus the Department of Health and Social Security (DHSS), and Supplementary Benefit legislation in particular. (The DHSS is a natural choice for such a project. It is a notoriously large and complex organisation, often said to be the largest administrative organisation in Western Europe. And within the DHSS, the administration of Supplementary Benefit law accounts for a substantial amount of resources.) The Logic Programming Group at Imperial College is one of the collaborators in this Demonstrator project. The other collaborators are ICL, Logica, and the Universities of Lancaster and Surrey.

The DHSS Demonstrator project has several aims, including the production of a system to support the clerk who must assess claims for Supplementary Benefit, and the production of another system to advise claimants on their likely entitlement to Supplementary Benefit and other kinds of benefit available in the United Kingdom. We are not contributing directly to either of these systems at this stage, but to another aspect of the Demonstrator (see [3]). However, because of our involvement with the Demonstrator, it was natural for us to choose Supplementary Benefit legislation also. For one thing, the choice of Supplementary Benefit meant that our approach could be compared with that taken on the Demonstrator, and that there would be an opportunity for cross-fertilisation.

Supplementary Benifit legislation

The United Kingdom's Supplementary Benefit legislation (at present) is contained in the Supplementary Benefits Act (1976), supplemented by ten sets of regulations. Regulations are statutory instruments made under enabling provisions contained in the act, and tend to contain more detailed definitions of the relevant concepts. These constituent parts are collected together into 'The Yellow Book' [9] which consists of some 600 pages.

Responsibility for the administration of the Supplementary Benefit legislation belongs to the DHSS. On a day-to-day basis, Supplementary Benefit law is applied by clerks of the DHSS who have no formal legal training. The DHSS supplies these clerks with an official handbook containing detailed guidance on how the law may be interpreted, together with information as to the procedures which the clerk must perform when assessing entitlement. This handbook is called the 'S-Manual' and is of a size comparable to the original legislation itself. The S-Manual could be viewed as containing, amongst other things, an alternative version of the knowledge contained in the legislation, and a condensation of the relevant case law. The existence of the S-Manual therefore provides an opportunity to give an extra dimension to the project. It allows us to compare and contrast formalisations of the law taken from different sources of the same domain knowledge. In addition, though this was not one of the aims of our project, there is in principle the possibility of investigating whether the S-Manual is consistent with Parliamentary legislation. Since the two are amended and maintained separately, there is no guarantee that they are consistent.

Others have looked at Supplementary Benefit legislation. Of particular relevance is the previous work by Hammond [5] and Leith [10].

USES OF A FORMALISATION

At this point it is necessary to elaborate on our view of the relationship between a formalisation of a piece of legislation and a system which would support a task related to that legislation. There are typically many different tasks which are performed on the basis of the same piece of legislation. Much of the Supplementary Benefit legislation, for example, lays down conditions under which the Secretary of State can pass supplementary regulations, and the circumstances in which he has discretion. It also imposes various duties such as

> "It shall be the duty of the Secretary of State to provide and maintain places, to be known as resettlement units, at which persons without a settled way of life are afforded temporary board and lodging with a view to influencing them to lead a more settled life."

For the purposes of our project, many of these provisions are not directly relevant. We chose to address only the parts of the legislation which contribute to the determination of an individual's entitlement to Supplementary Benefit

Nevertheless, even the definition of entitlement to Supplementary Benefit is used for different purposes. For example, the same legislation is used both by the clerk assessing entitlement and by people who advise potential claimants on their likely entitlements. The requirements for a system are different in the two cases. This is evident because the information available to the users of the two systems will necessarily be different. A system which is intended to be used by clerks who assess claims will have to contain many supplementary regulations, ranging from the specific procedures to be followed when assessing a claim, to the various local guidelines which have to be taken into account. Examples will be given later in the paper. A system which advises on entitlement would not require such detail (and even if it did, it is unlikely that this kind of detail would always be made available). Instead, the advisory system would require a broader coverage of the legislation, and would need to include supplementary information representing, for example, what seems to be the current attitude of the DHSS to various questions, what kinds of claims have been successful in the past, and some indication of the chances that a particular claimant has of having a claim accepted.

Similar considerations apply in all areas of law: there are always different requirements for those who must apply the law and those who advise on how the law is likely to be applied in a given case. For any practical system it is clear that we will need more than what is stated in the legislation. We need in addition information about how to apply the legislation, and the information that will be available to those who apply it. One possible approach to building a practical system might be to ignore the legislation altogether and instead represent a combination of the problem solving and domain knowledge. This the approach that was taken in Hammond's Supplementary Benefit system which was based on an expert's knowledge of the legislation and how he applied it in practice. Similarly collaborators on the Demonstrator, who are constructing systems for assessing Supplementary Benefit and advising on Supplementary Benefit, are taking as their starting points the S-Manual and advice workers' handbooks respectively.

We could expect, however, that two systems based on the same legislation would contain a certain amount of duplication, and that at the core of both systems there could be a single shared representation of the relevant legislation. An *idealisation* of our approach is that we want to move from the legislation to a formalisation which is neutral as to how it is used. This formalisation would be executable, but it would not constitute in itself a usable system. If we wanted to go on to build a useful system, we would have to add to the formalisation some additional material to tailor the system towards the target user. Additionally, where the system can support a variety of tasks, it might be possible to have several sets of this tailored material, each of which is added to the original formalisation to produce a system for the specified user. Proceeding from a neutral formalisation has several attractions. Not only does it permit the re-use of the formalisation, but the system which results ought to be more structured and hence easier to maintain.

There are two ways in which a formalisation must be tailored. We will have to add supplementary information so as put the definitions of the bottom level legal concepts in terms that the user can be expected to know about, and to express the procedures that the user must follow. We must also consider the way the system is intended to behave. For example, our prototype formalisations are excecuted by APES [6], a PROLOG system which is augmented to generate dialogues with the user automatically, and to provide proofs of conclusions reached. APES works (like PROLOG) by reasoning backwards from a conclusion we want to establish, to data we have available. For some applications such as an advice giving system, perhaps, it is often more appropriate to reason forwards instead, looking for conclusions we can draw from the available data. Of particular interest in the assessment task is a system which does not assess, but which supervises the clerk's assessment. Such a system could be seen as "looking over the shoulder of the user". A detailed description of such a system is outside the scope of this paper: in general terms it can be constructed by viewing the assessment process as the construction of a database which represents the clerk's assessment of a claim, and the rules representing the legislation as integrity constraints on this database, rather than as deductive rules. The intended behaviour of such a system would then be to draw attention to disagreements between the clerk's assessment and the system's conclusions.The resolution of such conflicts is left to the user, however. There are at Imperial College several logic programming "shells" under development which are intended to support alternative modes of use. The project will provide an opportunity to test out these propotype shells on a substantial application.

The picture is idealised: it is necessary to have a substantial formalisation of real legislation to see the extent to which it is borne out by practice. It could be that in practice adding the supplementary material departs from the central shared formalisation to such an extent that we might as well have constructed two separate formalisations in the first place.

THE PROJECT

Our aim in this project was not, at this stage ,to build a practical system. Other collaborators are doing this. Rather we had two separate, and largely incompatible aims. On the one hand, we wanted to support our longer-term work on knowledge representation. We wanted to confirm our expectations of what problems of representation we would encounter, and to identify other problematic constructs for longer and more detailed consideration later. We discuss several such problems in later sections on knowledge representation. On the other hand, we also wanted to build a complete formalisation, even if it meant adopting short-term solutions to do so. A relatively complete formalisation was necessary if we were to investigate the practical possibility of building several different applications from one neutral representation of the legislation. It was also necessary because we wanted to address a number of other issues, concerned largely with problems of scale. Our aims in this respect were as follows.

Methodology We would like to establish some kind of methodology for the formalisation of legislation. Currently, every such project is approached on its own terms. We wanted to get more experience of formalising legislation and the problems that arise, and to increase confidence in the basic feasibility of the approach. We hoped to get from the project an idea of how best to approach the task of formalising regulations. Should one attempt to understand the whole set of regulations and their interactions before one begins coding, or should one plunge straight in and follow a trial and error procedure? What sorts of predicates should one construct, and what level of detail should one try to represent? For the longer term, if such applications are ever to be constructed on a fairly routine basis, it will be necessary to develop techniques for monitoring and auditing the formalisation as it evolves. For our project, the accuracy of the representation was not a critical consideration at this stage. Our formalisation could therefore be undertaken with no expert legal assistance (except that the S-Manual does provide some indication of how to read the various provisions). In general, accuracy of the formalisation is, of course, critical, particularly if one were constructing a representation to be used in practice.

Resources At present there are no clear ideas of the resources needed to undertake this kind of work. There is little understanding of how long a large-scale formalisation might take, or of how to update and maintain a legal knowledge base as the number of rules increases. We felt that a large formalisation would help to improve understanding in these areas. We argued in describing our British Nationality Act project that a large scale formalisation will inevitably require frequent revision and adjustment. We hoped to get some feel for the sort of programming environments that would assist in the process, and possibly to compile a "shopping list" of desirable features. More generally, what software tools would be useful for a person coding legislation?

Entitlement to Supplementary Benefit

The aspect of the legislation on which we focussed is the definition of entitlement to Supplementary Benefit. Section 1(1) of the Act gives the basic definition of entitlement:

> "1.- (1) Subject to the provisions of this Act, every person in Great Britain of or over the age of 16 whose resources are insufficient to meet his requirements shall be entitled to benefit as follows-
>
> (a) a supplementary pension if he is one of a married or unmarried couple of whom one is or both are over the age of 65 or if he is not one of such a couple and has attained pensionable age; and
>
> (b) a supplementary allowance in any other case;
>
> and to such benefit by way of a single payment to meet an exceptional need...."

Other sections of the Act consist of enabling provisions which allow that elsewhere there may be exceptions to the simple definition. For example, while above it is stated that usually a person has to be in Great Britain in order to be entitled to benefit, section (1A) says

> "Regulations may provide for a person's entitlement...
> to continue during prescribed periods of the person's
> temporary absence from Great Britain."

Regulations are statutory instruments made under enabling provisions contained in the Act. In a set of regulations called *Conditions of Entitlement Regulations* there is one called *Persons abroad whose entitlement is to continue* which gives in detail the circumstances under which an individual can still be entitled to Supplementary Benefit even though absent from Great Britain.

In fact, the major part of the definition of entitlement to benefit is to be found in such regulations, particularly in the *Conditions of Entitlement Regulations, Resources Regulations, Requirements Regulations* and the *Aggregation Regulations.*

The parts of the Act that deal with entitlement to Supplementary Benefit proper, together with the relevant supplementary regulations, make up about 120 pages, or a fifth of the Yellow Book.

State of the implementation

The first version of the program was written (by two previously inexperienced people) in around two months. It runs to some 450 clauses and can be executed in the APES [6] either on a SUN workstation or on a personal computer. It is yet to be embedded within the other "shells" referred to above. We estimate that approximately 90% of the relevant legislation was covered.

This would seem to confirm previous experience that the 'prototyping' of such a formalisation is rapid. However, such statistics are largely meaningless. In this particular case, the legislation splits naturally into separate and fairly self-contained parts ("requirements" and "resources", for example) which could be worked on in parallel. This will not always be so. Further, in constructing a "complete formalization" some knowledge representation problems will always be passed over, and these might well eat up resources were they to be tackled. We give some specific examples from our formalisation of the Supplementary Benefit legislation later. Finally, though we have no reason to think so, it could be that the lack of expert guidance made our task appear easier than it really was, since some problems may well have been passed over unnoticed.

METHODOLOGY

The approach we took in constructing our formalisation was a top-down one, essentially as described in [11]. We were interested in a definition of entitlement to benefit in terms of lower level concepts. Concepts that are undefined in the legislation (such as "hard to heat") were treated either as data to be supplied by the user, or as qualifications to conclusions drawn by the system. However, where it was clear that common sense definitions could be given in terms of such predicates as "age" and "sex", we gave them.

This top-down approach contrasts directly with approaches that attempt some preliminary analysis of the legislation to identify the entities and relationships that appear. The best known example of this different approach is Stamper's LEGOL [13]. For a more detailed comparison between the two approaches, see [11].

Intermediate representation

It can be argued that we formalized too early, that we should aim to produce in the first instance a complete *intermediate representation* of the legislation, one that represents the complexities and detail found in the legislation, and yet is at a higher level than the executable formalisation.

There are a number of advantages to such an approach. It would impose some discipline on the activity, making it less of a trial and error process, and would perhaps make the writing of executable clauses almost a routine task. It would have the virtue of making more accessible our understanding (our interpretation) of the legislation and so could be used in verifying the accuracy of the formalisation. An intermediate representation would be useful too if the eventual implementation language of the formalisation was not decided on.

We did not use an intermediate representation. We did experiment with representational formalisms such as structured English and various forms of networks. However, the form of our executable representation is already so close to structured English that we came to feel that we were doubling our workload unneccesarily by constructing an intermediate representation in addition.

We remain aware of the advantages that an appropriate intermediate representation would provide. We also believe that it is worth investigating the extent to which a more systematic method of analysis (developments of Stamper's more recent LEGOL methodology, perhaps) could be combined with our pragmatic approach.

Structure of the Legislation

In the case of Supplementary Benefit legislation, the complexity of the legislation itself is a major problem. There are two aspects to this.

First, the fact that the legislation is distributed across an Act and various sets of regulations can make it extremely difficult to co-ordinate clauses that relate to the same concept. Although in many cases a definition does provide explicit references to other fragments which contain exceptions or amendments to that definition, it is not always so straightforward. For example in Section 1 (3) of the Act we find:

> "The requirements of any person to be taken into account for the purposes of this Act do not include any medical, surgical, optical, aural or dental requirements; and regulations may provide that the requirements which by virtue of this subsection are not included in a person's requirements include or exclude prescribed requirements."

The first half of this clause appears to state the following:

```
requirements to be taken into account =
      total requirements - medical etc. requirements
```

But the second half of the clause states that regulations may make this equation untrue - that they can include or exclude, with no bound, items not to be taken into account as requirements. The purpose of this second clause is to enable legislators to alter what is to be taken into account when calculating a person's requirements. In this case, supplementary regulations do not adjust the general rule with minor exceptions; rather, they might override the general rule completely. In this example, the most sensible way of proceeding would appear to be to ignore the provision entirely and use only the supplementary regulations to define "total requirements to be taken into account". The problem is that we have no way of determining whether there are any supplementary regulations without looking through the entire volume of legislation.

Secondly, the legislation contains an enormous amount of cross-referencing which often creates complex chains of reference difficult to understand. For example, regulation 9 of the *Conditions of Entitlement* regulations defines what it is to be in remunerative full-time work. For this, it may be necessary to determine whether a claimant:

> "has received..earnings..which by virtue of regulation 9(2) of those regulations (the Resources Regulations) as modified by paragraph (3) of this regulation (regulation 9) fall to be taken into account for a period subsequent to the termination of the employment..."

When one examines regulation 9(2) of the Resources Regulations, one finds that it begins

> "Earnings and other income shall be calculated on a weekly basis and, except in so far as regulations 3(2)(d)(i) and 13 provide otherwise..."

The chain does not end there, because regulation 13 begins:

> "Any periodical payment, including any arrears paid periodically, and, subject to paragraph (4)(a)..."

Needless to say, paragraph 4(a) includes:

> "and the lump sum payment shall, notwithstanding regulation 3(2)(b),..."

We shall consider the definition of "remunerative full-time work" in more detail later, in the section *References to the text of the legislation.*

With hindsight, we now believe that a promising way of dealing with this phenomenon would be to construct a database representing the structure of the legislation. Such a database might have uses other than that of helping us to analyse and keep track of such an arcane structure. It may help with updating and maintaining code as the legislation is amended. It could also be incorporated in a system to aid explanations. We have therefore recently begun investigations into what kinds of information such a database should contain, though the work is too preliminary to report on here.

The remainder of this paper deals with various aspects of knowledge representation. We begin, however, by considering a very general question: what is the appropriate level of detail we should attempt to represent? The question is so general that we address it in this section on methodology.

Level of Detail

A point that must be considered when representing the legislation is the level of detail we wish to represent. This is a question that a methodology should address because it has proved to be a persistent and puzzling one: *what sort of predicates should our formalisation include?* Consider the following from *Conditions of Entitlement* Regulation 6 (b):

> [a claimant is not required to be available for employment if] "...he is regularly and substantially engaged in caring for a severely disabled person..."

This could be formalised in several ways. A first possibility is to put the whole condition into a unary predicate name:

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

Obviously this hyphenated predicate could be decomposed further. One could, for example, introduce two relations and include a parameter to represent the subject of X's caring:

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

The question is whether such a decomposition would be useful. This question cannot be answered by consideration of Regulation 6b in isolation - the decomposition would be useful only if these two subordinate predicates were used elsewhere in the legislation. If the two concepts always occur together, nothing is lost in retaining the single large predicate. Further decompositions might be suggested by other parts of the legislation. If other clauses referred to other degrees of disablement, we would have something to gain by having a predicate 'disabled' with a parameter for degree.

Others have wished to take a quite different approach, wanting some kind of deep model in terms of which we could represent disablement and its degrees. But, even leaving aside the practicality of constructing such a model, there would be little to be gained in this case. This is particularly apparent when we refer to the S-Manual for an explication of the predicate concerned. There we find that a severely disabled person is simply defined as a person who is entitled to Attendance Allowance (another Social Security Benefit), or who fulfills the criteria for Supplementary Benefit attendance expenses. Thus, in practice, all the clerk need do to decide whether a person is severely disabled for the purposes of 6b is to consult that person's benefit history. It would be quite different for the advice worker, of course, who might have no access to this kind of detail. Considerations such as these provide a powerful argument for using the hyphenated predicate; it can be decomposed in different ways according to the application that is to use it. This is not to say that some kind of "deep model" of the underlying domain is not desirable. Some representation of the domain is essential. We would normally want to represent, for example, that a severely disabled person, or a person who is blind, cannot be caring for another severely disabled person.

Our guiding principle is to use the least amount of detail that is appropriate. What is appropriate will depend largely on the rest of the legislation. Further decomposition will be required for the practical systems, but this decomposition will be specific to the system, and is likely to differ according to the particular system.

KNOWLEDGE REPRESENTATION ISSUES

The remainder of the paper is concerned with specific problems of knowledge representation. From this point of view, there are essentially two features which distinguish the Supplementary Benefit legislation from other examples we have considered: the demands imposed by the legislation on the representation of temporal information; and the heavy reliance on explicit references to the text of the legislation.

Representation of time

To deal with the subject matter of law it is often necessary to represent and reason about the way objects and relationships change in time through the influence of events that occur in the world. For example, an individual's entitlement to Supplementary Benefit might change, as he or she falls ill, or becomes unemployed, or retires. The event calculus [8] was developed to formalise and implement reasoning about the effects of such events within a logic programming framework. The representation of legal rules and regulation provides an ideal domain in which to test and develop the event calculus.

We expected that among the most difficult Supplementary Benefit regulations to deal with would be those involving time. There are several reasons for this. Our British Nationality Act program was not constructed using the events calculus, but there the temporal aspects of the domain are simple: to determine if an individual is a British citizen at a given time, we need consider only events of acquisition of British citizenship (such as birth in certain circumstances, or naturalization and registration) and events of termination (such as renunciation of citizenship, or death). Our British Nationality Act program was constructed in the spirit of the event calculus, but we felt that the temporal reasoning required in that application would benefit little from importing the general mechanisms of the event calculus.

In contrast, to apply the Supplementary Benefit legislation does require a substantial amount of temporal reasoning. In particular, many of the Supplementary Benefit regulations require that we know, not only whether an individual satisfies a given condition at a particular time, but also the duration of time for which this condition has been satisfied. For example, an individual might be entitled to the long-term scale rate of benefit, but only if he has been entitled to Supplementary Benefit continuously for more than a year. There are so many such conditions in the Supplementary Benefit legislation that importing some general mechanism is worthwhile.

Although we did not employ the event calculus directly at this stage, (adopting instead special purpose versions of it as required), we have identified a number of constructs which are problematic even for the event calculus.

One source of these problems is the fact that Supplementary Benefit legislation (in common with much of the United Kingdom's welfare law) specifies its own idiosyncratic definition of what constitutes a continuous period of time. Moreover, the Supplementary Benefit legislation does not have a single notion of such a period, but defines explicitly what should be understood in particular contexts.

For example:

> "...where a person has been a patient for two or more distinct periods separated by one or more intervals each not exceeding twenty-eight days, he shall be treated as having been a patient continuously for a period equal in duration to the total of those distinct periods."

It is possible to represent such provisions in the event calculus, by adding to the general axioms for dealing with continuous periods of time, special purpose axioms which represent the detailed requirements of provisions such as the one quoted above. This kind of modification to the temporal model underlying the event calculus is relatively straightforward, because the event calculus states explicitly its temporal reasoning axioms, rather than treating them implicitly, as happens, for example in most special purpose logics based on modal temporal operators.

A second source of difficulty arises because many of the Supplementary Benefit regulations are retrospective. The formalisation of retrospective provisions requires dealing not only with changing states of objects in the world but also with changing states of knowledge. For example, in one state of knowledge, say at t1, it might be deemed that an individual, John, was not entitled to benefit at t0. However, in a later state, say t2, it might be decided that he was entitled at t0, for example if he successfully appeals against the decision. In other words, at t1 it is not the case that John was entitled to benefit at t0 whereas at t2 he was entitled to benefit at t0.

The formalisation of changing states of knowledge can be accomplished through the implementation of metalevel reasoning. For example, it is important to distinguish the event of John's falling sick from the event of *discovering* that such an event took place. The times of the two might coincide, but normally they will not. At the object level, we reason about events like John's falling sick to conclude that John was entitled to benefit for some period of time. At the metalevel, we reason about what we know, about events (like discovering that John has fallen sick) that change our state of knowledge. Separating the two kinds of event allows us to distingush, for example, the time at which John began being entitled to benefit from the time at which we began to believe that John was entitled to benefit.

The representation of time required by the Supplementary Benefit legislation demands sophisicated reasoning techniques. The event calculus is currently being developed further to address such issues as those mentioned above. Further discussion is beyond the scope of this paper, however.

Constructs Referring To Other Sections

In the case of the British Nationality Act, a major source of problems concerned representing adequately references to the text of other sections. In the British Nationality Act, many of these problematic references resemble "counterfactual conditions". This example was given in the paper that describes the work:

> "... became a British citizen by descent or would have done so but for his having died or ceased to be a citizen ... [by] renunciation."

This kind of "counterfactual" is discussed in more detail elsewhere [2], where the highly ambiguous nature of such constructs is stressed, and where it is argued that their meaning must be construed in terms of implicit references to other sections of the legislation.

Explicit reference to other sections of the legislation can also be a source of problems, however. The next section provides some specific examples from the Supplementary Benefit legislation.

It is the relative absence of references to the text of the legislation in the British Nationality Act which led us to say that it is an unusually well-written piece of legislation. The Supplementary Benefit legislation, in contrast, is large, heavily amended, and contains an enormous number of problematic references.

One particular kind of explicit reference is found in what might be termed "catch all" clauses. It is quite common to find in the Supplementary Benefit legislation a long list of conditions terminated by a phrase such as

> "or the previous paragraphs do not apply to him, but the circumstances are analagous to any circumstances mentioned in one or more of those paragraphs."

In the case of Supplementary Benefit there is a good reason why we can expect a large number of such constructs. Supplementary Benefit is the "safety net" of the UK's welfare legislation. It is the benefit that a person receives if no other benefit covers his needs.

We discuss in the following sections some of the problems that can be encountered. We remark here, in passing, that "catch-all" clauses, though apparently problematic, are easy (if tedious) to deal with in practice. In particular, there is no need to construct some kind of "analogy engine" to treat them. "Analogous to" in this context is merely a vague term which gives the assessment officer a certain amount of discretion. But interpretation of this vague term is constrained by precedent and case law in precisely the same way as interpretation of any other vague term in law is constrained.

REFERENCES TO THE TEXT OF LEGISLATION

In our formalisation of the Supplementary benefit legislation, and indeed in all other fragments of legislation we have considered, it has often proved convenient to introduce predicate names which refer to a section of the legislation.Often such predicates are introduced merely for convenience, to structure the rules in the formalisation. For example, a rule like:

> CE 6. "A claimant shall not be required to be available for employment [if].. one or more of the following paragraphs apply..:
> (a) he is not a partner..
> (b) he is regularly and substantially engaged in caring for a severely disabled person..."

could be represented directly as:

```
X shall not be required to be available for employment   if
  [X  not a partner
or
  X is regularly and substantially engaged in caring for a
  severely disabled person]
```

Alternatively, we might choose to eliminate the disjunction in the conditions of this rule by introducing instead an auxiliary predicate:

```
X shall not be required to be available for employment if
  X satisfies CE reg 6
```

where

```
X satisfies CE reg 6  if
  X not a partner

X satisfies CE reg 6 if
  X is regularly and substantially engaged in caring for a
  severely disabled person
```

Here, the predicate name "satisfies CE reg 6" has no particular significance.We need an auxiliary predicate. Rather than choose one arbitrarily, it is natural to use one that refers to some appropriate section of the text.

In heavily cross-referenced legislation like the Supplementary Benefit legislation, this usage takes on greater signficance, for the draftsmen themselves often refer to section names explicitly. For example:

> CE 6 (s):
> "he is a person to whom either of the following applies:
> (i) regulation 9(2)(a)..
> (ii) regulation 11(a), (b), (c) or (d)"

We also use predicates names which refer to the legislation when formalising other sorts of cross-references, in particular when exceptions to general rules in one part of the text are expressed in another:

> CE 7 (1):
> "*Subject to regulation 8*, a claimant shall be treated as available for employment.."

can normally be represented as:

```
X available for employment  if
  [P]  and
  NOT X excluded by regulation 8
```

X excluded by regulation 8 if....

Here NOT is treated by negation as failure.

In summary, including references to sections of the legislation in predicate names (or as arguments in more general predicates) often provides a convenient and natural way to express what the legislation seems to specify. Nevertheless, this technique does not eliminate all the difficulties, nor the need to perform a thorough analysis. Consider, for example, Regulation 9 (1) (a) of the *Conditions of Entitlement* , which defines what it is to be in remunerative full-time work:

> "a claimant is engaged in remunerative full-time work only where...subject to paragraph (2)...he is working for at least 35 hours a week..."

One straightforward formalisation of this might be:

```
X  is-engaged-in-remunerative-full-time-work   if
  NOT X satisfies 9 (2)  and
  X is working for at least 35 hours a week...
```

Paragraph (2) is satisfied if any of its five sub-paragraphs are satisfied, one of which is 9 (2) (b):

> "paragraph (1)(a) shall not apply to a claimant...who is working within the meaning of (1)(a) as a self-employed person..."

One straightforward formalisation of this might be:

```
X satisfies 9 (2) (b)  if
  X is-engaged-in-remunerative-full-time-work and
  X is self-employed...
```

If we add a rule saying that someone satisfies 9 (2) if he satisfies any of its sub-paragraphs, then we have a circular definition of being in remunerative full-time work. From a computational point of view, we have a program which loops. It is not only the computational point which is important, however. For if we now make explicit the assumption that there are no other ways of being in remunerative full-time work and no other ways in which a person can satisfy the conditions of section 9(2)(b), then it is a straightforward consequence of our formalisation that

```
X is-engaged-in-remunerative-full-time-work if
  X is working for at least 35 hours a week... and
  NOT X is self-employed ... and
  NOT X satisfies some section of 9(2) other than
  9(2)(b)
```

This transformation eliminates the circularity in the definition (and the loop in its execution as a program). However, it is now also a simple consequence of the formalisation that no-one can possibly satisfy section 9(2)(b).

We might argue on this basis that the phrase 'working within the meaning of (1) (a)' cannot refer to the whole of section 1(a), but only to those conditions which remain if the exception is ignored. This is reasonable in the context of section 9 (2) (b), but later, the phrase 'working within the meaning of (1) (a)' is used in a different context which suggests that now it is the whole of section 1(a) that is intended. We are left, therefore, with no alternative but to introduce two predicates for the two different readings of 'working within the meaning of (1) (a)'.

This kind of analysis is typical of that which must be undertaken as the formalisation proceeds. Heavy use of cross-referencing in the legislation makes such analysis extremely tedious. Worse, we cannot guarantee that we have analyzed all the provision of the Act to this level of detail. In other words, we might have overlooked other, similar problems, elsewhere in the legislation. This possibility is one reason why we suggested earlier that it is difficult to claim one has covered the whole of a piece of legislation.

There is a possibility that we might be able to automate parts of the analysis. Certainly, the problem in the example given above can be identified by applying standard syntactic tests to the formalisation. Such tests do not indicate, however, how any problems identified should be resolved. Moreover, as the examples in the following section will illustrate, a general technique could not be expected to cope with all the problems that can arise.

DEEMING PROVISIONS

A common device used in legislation is what might be called "the deeming provision". A simple example from the Supplementary Benefit Act is the following:

> " a person shall be treated as blind if he regained his
> eyesight within the previous six months at a time when
> a pension or allowance was payable in respect of him."

This particular example can be treated straightforwardly. For example, we could replace all occurrences of the predicate Blind by Blind*, where Blind* is defined as:

```
Blind*(X) if  Blind(X)
Blind*(X) if  RegainedSightEtc(X)
```

Deeming provisions often introduce negative conditions. For example:

> Resources Regulation 8(1):
> " Where a claimant's capital resources ...
> (a) exceed ...[£3000]... but
> (b) would be reduced to or below that sum if the capital resources of a dependant were disregarded,
> the capital resources of that dependant shall be disregarded as a capital resource..."

might be formalised:

```
CapitalResources* (x  z) if
        CapitalResources(x  z) and
        not [z > 3000 and
             DependantCapitalResources(x  z1) and
             z - z1 ≤ 3000]

CapitalResources* (x  y) if
        CapitalResources(x  z) and
        z > 3000 and
        DependantCapitalResources(x  z1) and
        z - z1 ≤ 3000 and
        y = z - z1
```

Here the additional negative condition in the first clause, not mentioned explicitly in the legislation, is required because we know that a person can have only one value for capital resources. Notice that if the formalisation is to be executed in PROLOG, then including additional negative conditions can be avoided, by writing specific clauses before general ones, and relying on PROLOG's search rule and the use of "cut" to remove the non-determinism. This is not an adequate solution, however. It works in PROLOG, but destroys any declarative reading that the formalisation has. Further, it precludes any possibility of using an alternative execution strategy on the formalisation.

Simple examples of deeming provisions, such as those given above, are tedious, but cause little conceptual difficulty in practice. We might be tempted, therefore, to devise some general mechanism which would eliminate much of the tedium of dealing with such provisions. This might take the form of a programming tool which automatically modifies predicate names as appropriate, and introduces any additional negative conditions that are implied. Alternatively, it would be straightforward to modify the interpreter so that it could handle such deeming provisions directly.

However, consideration of a wider range of examples shows that any such general attempt at treating deeming provisions will be inadequate to deal with all cases. Consider, for example, the following regulation which specifies how to calculate part of a person's resources:

> Regulation 3 (c) (Resources regulations):
> "any instalment of a capital payment..shall be treated-
>
> i) in a case where, if the instalment were treated as a capital resource, the amount of the claimant's capital resources would not exceed (£3,000...), as an income resource,
>
> (ii) in any other case, as a capital resource.

Suppose now that a person has two payments, A and B, and that the amount of capital resources about which there is no doubt is T. If c(A) denotes the contribution of payment A to the total capital resources, then regulation 3(c) tells us that

$$\begin{array}{lll} c(A) = 0 & \text{if} & T + A + c(B) > 3000 \\ c(A) = A & \text{if} & T + A + c(B) \leq 3000 \end{array}$$

Similarly,

$$\begin{array}{lll} c(B) = 0 & \text{if} & T + B + c(A) > 3000 \\ c(B) = B & \text{if} & T + B + c(A) \leq 3000 \end{array}$$

There is no difficulty in applying this regulation if, for example, T is £2500, A is £500 and B is £600, say. Suppose, however, that T is £2000. Then there are four possible values for T+ c(A) + c(B) (the person's total capital resources) consistent with the regulation. The four possible values are £2000, £2500, £2600, and £3100.

The S-Manual gives no further guidance in this case. In practice, apparently, the situation of two such payments A and B does not arise. The fact that the legislation has made no provision for such an eventuality, however, makes it difficult to represent the regulation faithfully, and still make use of it in a given practical application.

We conclude by remarking that the S-Manual does resolve a similar difficulty in the context of another regulation. There, the clerk is told to

> "Assess the case in the way most favourable to the claimant..."

This too could be regarded as a certain kind of deeming provision, yet it is difficult to see how any general mechanism for treating deeming provisions could be expected to handle this kind of self-referential statement. In our formalisation of the Supplementary Benefit legislation, we adopted ad hoc solutions to such problems as they were encountered.

We have recently embarked on a classification of the various kinds of deeming provision that appear in the Supplementary Benefit legislation. This is a first step towards investigating the possibility of providing a library of ready-made treatments for the simpler kind of deeming provisions.

CONCLUSION

We have described a formalisation of our interpretation of the conditions under which an individual is entitled to Supplementary Benefit. The significance of this work is not so much in the fact that our formalisation can be executed as a program. Of greater significance is the range of knowledge representation problems that we have encountered, and the experience we have gained in attempting a representation of a large and unwieldy fragment of legislation.

The problems of knowledge representation we encountered can usefully be divided into those concerning the representation of time, and those arising from heavy cross-referencing in the text of the legislation. It is these latter we feel that are characteristic of large fragments of legislation, in the sense that problems with representing time can arise anywhere.

We have firm plans on how we shall develop this work. We want to investigate the knowledge representation issues mentioned above. We have also identified a self-contained part of the legislation for further development. Further work will focus on the assessment of capital resources.

This development will take two forms. Where the regulations are particularly obscure, we shall formalise the corresponding part of the S-Manual, and then attempt to unhook the original clauses, and join the S-Manual clauses to the higher-level rules which remain. For the longer term verification of our picture (whether a formalisation can be used for several different tasks) we shall attempt to build a usable system for helping in the assessment of claims from our "neutral" representation. (More accurately, since we shall be working with a subset of the legislation, we shall be supporting only one aspect of the assessment process). In addition, this will enable us to address the representation of procedural law, something which accounts for large sections of the S-Manual, and something we have ignored so far.

Acknowledgement
This work was supported by the Science and Engineering Research Council.

REFERENCES

[1] Bench-Capon T., Sergot M. J., (1985) Towards a rule based representation of open texture in law. Dept. of Computing, Imperial College.

[2] Bench-Capon T.J.M., (1986) Counterfactual Conditionals and Logic Programs. Dept. of Computing, Imperial College.

[3] Bench-Capon T.J.M., (1987) Support for Policy makers: Formulating Legislation with the Aid of Logical Models. First International Conference on AI and the Law. Boston. May, 1987.

[4] DHSS Demonstrator (1984) A proposal to the Alvey Directorate.

[5] Hammond P. (1983) Representation of DHSS Regulations as a Logic Program. In *Proceedings of the 3rd BCS Expert Systems Conference*. British Computer Society, Cambridge, pp 225-235.

[6] Hammond P., Sergot M. J., (1983) A PROLOG shell for logic based expert systems. *In Proceedings of the 3rd BCS Expert Systems Conference*. British Computer Society, Cambridge, pp 95-104.

[7] Kowalski R.A. (1979) *Logic for Problem Solving* North Holland-Elsevier, New York.

[8] Kowalski R.A.and Sergot, M.J., (1986). A Logic Based Calculus of Events. *New Generation Computing*, Vol. 4, No. 1, Feb 1986, pp 67-95, Ohmsha Ltd. and Springer-Verlag.

[9] *The Law Relating to Supplementary Benefits and Family Income Supplements*. (1980). Her Majesty's Stationery Office, London.

[10] Leith P. ELI: An Expert Legislative Consultant. Presented at the IEE Conference on Man/Machine Systems UMIST July 6-9, 1982, Conference Publication Number 212.

[11] Sergot M. J. (1985) Representing Legislation as Logic Programs. Dept. of Computing, Imperial College. To appear in *Machine Intelligence 11*, Oxford University Press.

[12] Sergot M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., HammondP., Cory H. T. , (1986) The British Nationality Act as a Logic Program. *Comm. ACM*, Vol. 29, No. 5, pp 370-386.

[13] Stamper, R.K., (1979). LEGOL: Modelling Legal Rules By Computer. In *Computer Science and Law* (Niblett, ed.,) Cambridge University Press, 1979