9th BILETA Conference
Building Systems

1st & 2nd April 1993
John Moores University
Liverpool

Developing Multi-Problem Multi-Entity Advice Systems

Philip Boyd

Keywords: Social Security - law - legal information - correctness - advice welfare benefits - maintainability - efficiency - usability

Abstract: A system to determine entitlement to Social Security benefits for a family is typical of multi-problem/multi-entity development. Not only are there multiple entities: each individual member of the family and the family itself, but they are of different types: adult, child and family. Each entity is potentially entitled to a range of benefits which may or may not have an impact on the entitlement of other entities within the system, and sometimes on entities completely outside the system who are nevertheless closely related to the system entities in the real world. This paper seeks to describe some of the problems of developing and maintaining such a system, some techniques which the author has developed and used and some further techniques which the author is developing. It goes on to consider the isomorphic approach in this area and to consider the pursuit of correctness, maintainability, usability and efficiency.

1 Multi-problem / Multi-entity Advice Systems

Let us imagine four problems P1, P2, P3 and P4; the essentials for the solution of each are that:

P1 - a person must be a judge under 65 years old who reads the Guardian but there are different solutions for males and females and for car owners;

P2- a person must be married and over 60 years old but there are different solutions for males and male and female students and for Guardian readers;

P3 - a person must be a lecturer who reads the Guardian, but there are different solutions for males and females and for owner occupiers;

P4 - one member of the family owns a car and the family are owner occupiers, success or failure are the only solutions.

The task of the multi-problem/multi-entity advice system is to:

- estimate the likelihood of each solution to each problem for each of three individuals who make up a family;
- estimate the likelihood of the solution to the problem;
- correctly reflect the interconnection of the various solutions;
- allow the advisor to model differing individual and family circumstances and see the different problem outcomes.

2 Social Security Law

Social Security Law is typical of multi-problem/multi-entity environment. Each individual member of the family, and the family as a unit is potentially entitled to a range of benefits, which may or may not have an impact on the entitlement of other family members, the family as a whole or on other individuals outside the family.

Social Security Law is not a strictly defined cohesive body of law, rather it resembles 'a web spun by a drunken spider' (Briggs and Rees, 1980). It is statute based and subject to precedent but these precedents are not normally set in the Superior Courts. It is subject to the Common Law and influenced by the judgements of the European Court but most Social Security decisions are given by Inferior Tribunals. Thus, precedents are established by Social Security Commissioners rather than by senior judges and judgements which are contrary to the will of the government of the day are often nullified by changes in regulations, sometimes before the judgements are delivered.

Whilst Social Security Law cannot be said to be separate from general English Law, it is perhaps as separate as a body of law may be. Its other major attribute is its volatility which is partly due to its structure of widely drawn Statutes allowing Ministers to draw all the detail by means of Statutory Instrument, partly due to the pressure of precedent from Commissioners and the Courts and partly due to the visibility of the law in relation to visible groups of disadvantaged people.

2.1 Social Security Benefits

It has been said that the British welfare state cares for people from birth to death: it has also been said that with the introduction of family planning services and funeral grants, this care now extends from erection to resurrection. Whatever the truth of this, there are over seven different ways of getting what are broadly called 'welfare benefits' which range from legal help from high flying barristers to wigs and fabric supports, and from maternity grants to funeral grants.

Some benefits such as the Disability Living Allowance can be claimed by individuals of any age, some such as unemployment benefit can only be claimed by adults and some such as income support can be claimed only by 'families', although an individual can also be a family in his or her own right.

These benefits are administered by a wide range of central government departments and local authorities each making its own determination in isolation.

2.1.1 Benefit Administration in the real world

The agency receiving a claim will determine that claim on its merits, more often than not taking no account of other benefits to which the claimant is or may be entitled. Overlapping entitlement is dealt with at a later stage by a separate set of rules. For example, a man might be incapable of work and entitled to invalidity benefit with an increase in benefit for his wife. His wife might be unemployed and entitled to unemployment benefit in her own right. Both claims would be assessed on their respective merits and both benefits would be awarded but the overlapping benefits rules would prevent the payment of either the increase in invalidity benefit for the wife or the wife's unemployment benefit, although title to both would continue to exist.

Such a system can lead not only to confusion for claimants but also to a number of 'better off' situations in which families can be considerably better off (or worse off) depending on which benefits they claim and when and in what order they claim them.

Taken together with widespread ignorance of the many benefits available, the difficulties of claiming them and the interconnection of benefits one with another and with the income tax and national insurance system, the system of benefit administration gives rise to the need for independent advice for claimants.

2.1.2 Benefit Determination

The process of determining entitlement to any given benefit is a three stage process:

1. From the law which governs entitlement to this benefit, decide which facts need to befound;
2. Enquire about the facts;
3. Apply the law from (1) above to the facts from (2) above to determine entitlement.

The adjudicating authorities take a 'shotgun' approach to this process: they determine all the facts which need to be found in all possible cases, subject this list to some (very small) amount of filtering by consolidating facts which can only apply to a very small percentage of the claiming population and produce a form which asks all the common questions and all the filter questions with subsidiary forms to obtain further information when triggered by a filter question.

Since each benefit is viewed in isolation by those who administer it, there is no question of considering other entitlement or better off' tests except in very limited circumstances.

3 Social Security Advice

The social security adviser then, is not called upon to provide information and to give advice in one single domain but to be something of a generalist, not only having knowledge of some seventy different benefits, the rules of which change constantly, but also of the way they are interconnected, the advantages of working versus not working, and all the 'better off' situations which can arise.

There are probably no more than 400 people in the UK whose full-time job it is to give advice about social security. Many other people however are called upon for such advice as part of their day to day work: probation officers; social workers; CAB workers; and solicitors to name but a few. Such people often have a little knowledge but have precious little time to expand their knowledge nor keep it up to date. An efficient computerised advice system therefore allows them to give social security advice with some confidence in its correctness and to model with their clients a number of 'what if' situations to enable them to make better decisions about their lives.

3.1 Computerised Social Security Advice Systems

Essentially, a computerised Social Security system which represents the real world would not be difficult to construct, it would have a conversational intefface which would ask all the questions from all possible official forms without reference to their relevance, would draw all the appropriate logical and arithmetical inferences and produce a report on entitlement in the same way as a cross section of all possible adjudication officers would do. Such a system would not however be usable. Firstly, each consultation would take hours if not days and advisers would simply not be prepared to invest time on it. Secondly, the 'determination' section of the program would not automatically have any built in checks for correctness any more than theadjudication officer's brain has built in checks nor would the results be any more verifiable. Thirdly, there would be no automatic verification of the correct updating of both the determination section and the interface as no such automatic provision exists in the real world. Finally, the system would not know about the interconnection of benefits nor their connection with the tax and national insurance systems and would not be able to do the 'better off' calculations.

A computerised Social Security advice system then, should not simply represent the real world, it should:

- be selective, allowing the user to select areas in which the claimant is interested or has a likely

entitlement;
- be more efficient than the paper form, asking only those questions which are relevant to the interests and drcumstances of the claimant;
- have built in validation checks;
- be verifiable in some meaningful way;
- have some method of ensuring that changes forced by legislation or precedent are correctly reflected in both the questions asked and the 'determination' section;
- know about and be able to perform 'better off' and 'back to work' calculations and the ability to model the real world of the claimant in different circumstances.

Other desirable attributes would be:

- on-line information not only about how to answer questions but also about how to claim and how to overcome the difficulties of claiming and of challenging decisions;
- prompting about areas where the user has not declared an interest but which are likely to be relevant;
- different interfaces related to the different skill levels of those using the system and giving advice to claimants.

In summary then, we require a system which will deal with multiple entities: the family and each individual within it; deal with multiple problems: the entitlement of each entity to each benefit; deal with the interaction and interconnection of the results for each entity; enable the user to model different views of a claimant's circumstances in the same way as a manager might model a budget with a spreadsheet.

3.1.1 Maximiser Plus - Development

The Maximiser Plus suite of programs form a PC based advice system which I have developed over the last ten years. They are written in C and require a minimum of an 8086 processor, 400K available ram and 1 - 2 Mb hard disk space. It is theoretically possible to run the programs from floppy disks but the response times are so bad that it is no longer really practical.

The suite consists of four programs: Maximiser itself, which calculates entitlement to almost all welfare benefits for the family and each individual within it; Helper PC which deals only with the family and calculates entitlement to the major means tested benefits, In-Work PC which answers two questions - 'If I took this job at this wage, would I be better off?' and 'What gross wage would I need to earn to make it worth my while taking a job?' and XQ which allows the user to use the programs' help and information system and contains a library other materials.

The first work done in this area was a batch processing system developed in Scotland (Du Feu, 1975; Adler and Du Feu, 1975; Adler, Du Feu and Redpath, *1975-7)* and it was upon this basis that I developed a pilot program SECURE at the University of Birmingham in 1980-82. A parallel development was WRAP, developed by Cardiff CAB and it was this program which became the first commercial social security advice system to be marketed in the UK. WRAP was eventually bought by Ferret Information Systems Ltd of which I am a director. Major changes in both social security law and hardware in the mid 1980s forced a rewrite of WRAP to produce Maximiser I, a product which has been continuously developed since then.

The current programs provide a conversational interface asking all the relevant questions and no irrelevant questions. Each question has a default answer which is presented to the user for confirmation or alteration. The user is allowed to retrace his/her steps through the program and amend input data either before or after calculation. Finally, the program produces a screen report with the option of a printed report. Context sensitive, on-line help is available throughout the consultation and XQ provides the facility to view other relevant information to which the system has access.

3.1.2 Maximiser Plus - Structure

The program falls neatly into three distinct parts sharing one common data structure: the enquiry interface, the calculation section and the report section. Underlying these three parts is the on-line help system. Of these sections, the latter two are relatively simple.

3.1.2.1 Maximiser Plus - calculation Section

At first glance, it might be thought that the calculation section would be anything other than simple but, in the real world, determining entitlement to various benefits is not difficult once the relevant facts are established and is a task often undertaken by relatively junior staff. The most difficult arithmetical operation undertaken is to calculate a percentage, and most benefits involve no arithmetic at all being simply based on logic: If condition a and condition b and condition c and not condition d then entitled is true; if entitled is true then amount payable is *x*.

3.1.2.2 Maximiser Plus - Report Section

The only function of the report section is to report the conclusions of the calculation section to the screen and to a standard printer and is not difficult in terms of the problems and outcomes.

3.1.2.3 Maximiser Plus - User Interface and Question Selection Logic

The major development and maintenance task is in the user interface and the logic of questioning. It is essential that all the relevant questions are asked and that no irrelevant questions are asked. Further, it is essential to ensure that if an answer is changed, everything which depends upon that answer is reset and that all new possibilities opened up by the new answer are explored. For example, in the initial consultation, the user might say that the client is incapable of work. The program logic would then ignore any questions related to the claimant's earning power and would concentrate on the benefits payable to sick and disabled people. If, at a later stage, the incapacity answer were changed, the given answer to 'How many weeks has the claimant been incapable of work?' would have to be reset and new questions about the claimant's earning power would need to be asked.

The essential questioning logic for each entity's circumstances is overlaid by a further constraint imposed by the benefits or groups of benefits which the user has said are to be examined during this particular consultation.

Because most users are 'naive' computer users, the interface must also be foolproof: not only must each keystroke be validated but ranges have to be set which will cope with a very uncertain world. For example, when deciding the age range for which it would be appropriate to ask the question 'Is this person pregnant?' the advice which I received was that it should be 7-'65. People must be no less than 0 years old and no more than 110. Mothers must be at least 8 and their children must be at least 1 year younger.

Just in case the questioning logic fails to trap user errors, a validation section is required to pick up people who are said to work but to have no earnings and vice versa, people who are said to be incapable of work but have earnings etc.

When the law changes, not only do changes have to be made to the calculation section but questioning logic must be examined to ensure that questions which have now become irrelevant are removed and new questions are added and that these new questions are asked in (newly) relevant circumstances.

3.1.2.4 Maximiser Plus - On-line Help

The user on-line help system presents more of a management than a logic problem. As old questions disappear, their help units need to disappear with them and as new questions are added so new help units need to be incorporated. Tools are therefore required to ensure that all relevant and no irrelevant help is included. Of course it is essential to ensure that the help is up-to-date and reflects the current state of the law and the program.

3.1.3 Maximiser Plus - Development Problems

In the initial stages of Maximiser Plus, various system development tools were explored, notably Jackson System Design and the Entity - Relationship approach. Because of the peculiar nature of the problem, none was wholly successful. The approach from which most was taken was the Yourden System Design Approach but even that was not wholly appropriate. I have, over the years, consistently failed to identify whether it is one single factor in the problem, a combination of factors or my own incompetence which have rendered widely used system design tools useless for such a task and I invite comment on this problem.

Since Social Security appears to be a rule based system there is an immediate attraction to using an expert system shell as a vehicle for building an advice system. I therefore set out in 1986 to build such a system in ICL's ADVISER, a rule based system consisting of a domain independent inference engine driven by a goal seeking, backward chaining inference strategy.

At the end of the project I concluded that

> 'Despite the attraction of using a rule based shell to define a rule based problem I am bound to conclude that the final product delivers less than it initially promised in terms of development speed, maintainability and the user interface.' (Boyd, 1988)

The main practical difficulty with this development was the tension between the inference engine which wanted to ask questions in the most 'efficient' order and the users who wanted all the questions about money in one place and all the questions about health in another. A further major difficulty was the requirement to use 'meta-rules' to deal with filter questions such as 'Is any adult sick or disabled?' which avoided numerous further questions such as 'Can Fred walk?, Can John walk?, Can Anne walk?' etc. It was initially hoped that the knowledge base could be separated from the user interface and that development could proceed in parallel. This proved to be impossible because of the introduction of meta-rules, grouping rules and demands for a 'sensible' interface.

The results were that the 'purity' of the knowledge base was corrupted by interface requirements and the maintainability consequently reduced. A further factor reducing maintainability was the grouping of rules as a short cut. The temptation to do this was overwhelming but led to major re-coding when one set of rules changed in relation to the others. The overall level of maintainability proved neither higher nor lower than programs written in conventional languages.

As a consequence of this trial, a procedural language was chosen for the development of Maximiser Plus.

3.1.3.1 Maximiser Plus - Calculation Section

The development of calculation section produced few problems. Taking the first of our imaginary problems, the calculation section might look like this:

```
For each individual
if occupation = judge and age < 65 and guardian reader
```

```
  success
if success
  if male and car owner
     outcome 1
  if female and car owner
     outcome 2
  if male and not car owner
    outcome 3
  if female and not car owner
    outcome 4
else
  outcome 0
```

In computing terms, this is not a very difficult task, the main problem is verification. Formal methods are not suited to verifying such calculations, exhaustive testing is generally not possible largely because of the volatility of the subject matter and I would argue that the only reasonable method of verification is by inspection by another expert. But how can a non programmer inspect a piece of C code and say whether it correctly reflects the law?

3.1.3.2 Maximiser Plus - Report Section

Again, the development of the report section presented few problems. Taking our imaginary problems, the report section might look like this:

```
For each individual
  For each of problems P1 to P3
    Display problem outcome
  For problem P4
    Display problem outcome
```

3.1.3.3 Maximiser Plus - On-line Help

Whilst hugely complex in legal terms, in computing terms the development of the on-line help system produced few problems. Verification of the actual material proved problem free but verification that all required help was present, that no old and redundant help was included, and that each question called the correct help was much more difficult.

3.1.3.4 Maximiser Plus - Question Selection Logic

The development of this section was without doubt the most difficult task and the technique I found most useful was the 'variable or predicate bucket': this is simply a table with variables listed vertically, problems listed horizontally and the various relationships between them noted in the matrix.

Looking again at our four problems, the critical elements of P1 are 'judge', 'under 65', 'Guardian reader' and the non-critical elements are 'male', 'female', 'car owner'. These critical elements are ranked by rarity with the rarest being ranked first. The non-critical elements are ranked similarly. The other problems are similarly ranked.

	P1	P2	P3	P4
over 60		C1		
under 65	C3			

male	N2	N3	N2	
female	N3	N4	N3	
married		C2		
car owner	N1			C1
guardian reader	C2	N1	C2	
owner occupier			N1	C2
judge	C1			
student		N2		
lecturer			C1	

The first look at this table shows that it is not efficient and that we can collapse 'over 60' and 'under 65' into age; 'male' and 'female' into sex and 'judge', 'student' and 'lecturer' into 'occupation'.

	P1	P2	P3	P4
age	C3	C1		
sex	N2	N3	N2	
married		C2		
car owner	N1			C1
guardian reader	C2	N1	C2	
owner occupier			N1	C2
occupation	C1	N2	C1	

As a rule of thumb, we should then ask the questions according to first the combined critical ranking and then the combined non-critical ranking, which might give the following result.

	P1	P2	P3	P4
occupation	C1	N2	C1	
age	C3	C1		
car owner	N1			
guardian reader	C2	N1	C2	
married		C2		
owner occupier			N1	C2
sex	N2	N3	N2	

This technique is far from foolproof but nevertheless, at least it provides a starting point for this part of the development process.

3.1.4 Maximiser Plus - Maintenance Problems

The major maintenance problem has been concerned with 'linking'. The calculation, report and on-line help sections produce few problems other than those described in the development section. Restructuring of the question selection logic again proves the most difficult task and verification of correctness again proves impossible as does verification of the data validation section. Verification that all the required changes have been made to all parts of the program remains problematic.

4 Isomorphism as a way forward

It has been argued

> 'that knowledge based systems in the legal domain will be more disciplined and better engineered if the structure of the knowledge base reflects the structure of the source documents from which the knowledge base is derived'. (Bench-Capon, 1989)

and Bench-Capo,1 (1991) describes a system in which these principles are put into practice.

Hammond (1983) describes a project which seeks to represent benefit regulations as a logic program using APES (A PROLOG expert system shell). He concludes that 'DHSS regulations describing benefit entitlement are already in a form which is readily translated into logic programs' and 'The same logic program describing benefit entitlement could be provided with a different front end, chosen to suit users of varying sophistication'. It must be said however, that these systems deal only with iterations of one client with one problem.

Some, however, have doubts as to the value of isomorphism in single entity/single problem systems. In a description of an *Expert System to Determine Statutory Sick Pay,* (a welfare benefit administered directly by employers but based in essentially similar determinative law to DSS benefits), Sharpe (1985) notes that 'asking the user is all right if there is a good match between his real world concepts and the goals that the system must satisfy' and concludes that 'the match between the technique (of logic for rule based systems) and the domain exists only at surface level'.

Crossfield and Gilbert (1986) working on the DHSS Alvey Demonstrator Project conclude that

> 'we do not yet know very much about how to provide comprehensible explanations or advice, although it is generally recognised that the expert systems of the future must be able to give accounts of their actions if they are to be trusted in critical situations'

I would argue that it is clearly not sufficient to simply codify the rules about benefits from the source documents which emanate from Parliament, Commissioners and the Courts. Such a rule-base might be of immense help to the Social Security Law Expert, but of little value to the advice giver. To be effective, the rule-base must be built upon the rules under which the adviser operates rather than upon the benefit rules themselves. These rules must be a reasonable, cost/effective distillation of the benefit rules which produce reasonable advice on whether to claim a benefit for most people, most of the time. Furthermore, the knowledge base must contain or point towards not only the knowledge which the expert carries in his head but also the sources of information which he consults.

Because law itself is an abstraction, in dealing with representations of Law, one is seeking to represent reality at least one remove. Further difficulties with such representations are that:

- The law is ill constructed and subject to change not only by Statute and Regulation but also by precedent.
- Parallel changes may change the law in different directions.
- The doctrines of Ultra Vires and retrospective legislation mean that what was true yesterday may today and for all the past have been untrue.
- Simply updating the text of the law itself is difficult. Translating the law into computerised code is difficult. Updating that code is difficult.
- Mechanically verifying that this code produces the correct results is well nigh impossible.
- The option of beginning from the point of view of a logical information flow is not available

Yet another difficulty is that those who have so far succeeded in producing representations of the Law have found it necessary to become computer programmers of one kind or another and have become bogged down in data structures, memory management, display techniques and a thousand other things which have nothing to do with law.

5 Future Development

I propose that a step in the right direction of producing good advice tools for multi-problem, multi-entity advice systems is to develop and refine a structured methodology within which any branch of determinative law from any legal system can be analysed, specified and tested by the expert in that domain who need not necessarily have any knowledge of conventional programming and that the domain expert should be able, by means of the methodology, to demonstrate to his peers the correctness of his analysis.

I would suggest a 'toolkit' which will:

- assist the domain expert in this analysis and specification of the law;
- allow the domain expert to test his specification and to dernonstrate its correctness to his peers;
- provide a means to include his specification directly into the source code of a computer program without any changes by anyone else in any way;
- separate the 'what' of analysis and specification from the 'how' of programming whilst at the same time ensuring the correctness of the 'how' given the underlying correctness of the 'what';
- manage change within the analysis and specification correctly.

I suggest that techniques and tools are required which will enable the domain expert to separate the 'what' of data collection, data verification, calculation, help and information and result reporting; to assure himself that a change in the law is accurately reflected throughout the whole specification; and to ensure that the finished specification is directly included in the code given to the programmer who will concern himself with the 'how' of data collection, verification, calculation, help and information, and reporting.

I would suggest that the required tools might be:

- A Comprehensive Data Dictionary tying statute, regulations and precedent directly to elements of the Specification Language (q.v.).
- A Prioritisation Tool by which the domain expert can describe and prioritise rules.
- Compilation Tools by which the domain expert can 'compile' two or more sets of rules together to produce a logical order of data collection.
- A Specification Language - A logical, structured language, as simple as possible to use, which is both machine and environment independent and by means of which the domain expert can specify his rules and calculations.
- A Specification Interpreter by which the domain expert can test his specification with direct links to the Data Dictionary.
- A Code Generator by which the domain expert's specification can be directly included within the specific programming code for a particular machine or operating system.
- Data Dictionary Manipulation Tools by which change can be effectively and correctly managed.

References.

Adler M. and D.A. Du Fue (1975). *Computer-based Welfare Benefits Information System* IBM (UK) Ltd.

Adler M., D. Du Fue and A. Redpath (1975-7). *Welfare BenefIt Project Working Papers* University of Edinburgh.

Bench-Capon T.J.M. (1989). 'Deep Models, Normative Reasoning and Legal Expert Systems', *Proceedings of the Second International Conference on AI and Law,* Vancouver. ACM Press.

Bench-Capon T.J.M. (1991). 'Exploiting Isomorphism: Development of a KBS to support British Coal Insurance Claims', *Proceedings of the Third International Conference on AI and Law,* Oxford. ACM Press.

Boyd R.P. (1988). *WALLY - An Expert System for Welfare Benefit Assessment* University of Aston.

Briggs E. and A.M. Rees (1980). *Supplementary Benefit and the Consumer* Bedford Square Press.

Crossfield L.P. and G.N. Gilbert (1986). 'Introducing Expert Systems into a large legisIation-based organisation' in *Expert Systems and Knowledge Engineering* North Holland.

Du Fue D. (1975). *A Computer based Welfare Benefits Information System : System Documentation* IBM (UK) Ltd.

Hammond P. (1983). 'Representation of DHSS Regulations as a logic Program' *Department of Computing Report* Imperial College London.

Sharpe W.P. (1985). 'Logic Programming for the Law' in Bramer, M A (Ed) *Research and Development in Expert Systems* Cambridge University Press.