CENTRAL GOVERNMENT PENSION RULES AS A LOGIC PROGRAM

*

K.K. BAJAJ, R.K. DUBASH
KBCS NODAL CENTRE, DEPARTMENT OF ELECTRONICS,
LODI ROAD,A BLOCK, CGO COMPLEX,NEW DELHI
E-MAIL : +uunet!shakti!vikram!kbcs
AND
ROBERT KOWALSKI
DEPARTMENT OF COMPUTING, IMPERIAL COLLEGE OF SCIENCE & TECHNOLOGY, UNIVERSITY OF LONDON, LONDON, ENGLAND
E-MAIL : shakti!uunet!NSFNET-RELAY.AC.UK!doc.imperial.ac.uk!rak

Abstract

An automated legal reasoning system for the Central Civil Services (CCS) Pension Rules is under development. This paper discusses the use of logic programming for representing the knowledge contained in the rules and how the logic program can be used as an expert system. The emphasis of the paper is in the use of temporal reasoning in the laws under consideration, the separation of the logic part of the program from the user data interface and the interaction of the user through forms with the knowledge base. The importance of the interplay of the propositional logic analysis of the rules with the entity-relationship analysis for the determinmation of predicates and parameters is also discused.

1. Introduction

Logic programming is one of the knowledge representation paradigms. Although it has received considerable attention as a basic pillar for fifth generation computer system ever since the Japanese project was launched almost a decade ago, it is only recently that application development has picked up using logic programming for representing knowledge. In the area of automated legal reasoning systems, one of the first applications of logic programming was for the British Nationality Act (1). The group at Imperial College has used logic programming for other applications such as the social security disbursements under the Department of Health and social Security (DHSS) Program (2). Logic programming has been found to be an effective scheme for representing legal knowledge.

Logic programs represent knowledge in the form of statements

$$A \text{ if } B_1 \text{ and } \ldots B_n, \quad n \geq o$$

* Any correspondence may be addressed to K.K. Bajaj.

Where A and B are all atomic formulae. The conditions B_i can also be negative atomic formulae. Most knowledge can be represented in the form of such statements and hence by logic programs.

In this paper we present the use of logic programming for representing the knowledge contained in Central Civil Services (CCS) Pension Rules and how the logic program can be used as an expert system. A similar automated reasoning system for import policy legislation is also under development and has been reported elsewhere (3). The two applications present technical challenges in logic programming which are quite diverse in nature. While the latter has comparatively shallow reasoning power, wider breadth of data, larger data base, more complex user interface, relatively simpler English with fewer ambiguities in language, the former offers the possibilities of deeper reasoning, more complex English language, complex temporal reasoning. The applications demonstrate the power of logic programming in different domains of reasoning in law.

The CCS Pension Rules are applicable to most government employees. These rules determine the number of years which qualify for pension, regulate the amounts of pensions, decide on the classes of pensions and conditions governing their grant. Determination and authorisation of the amounts of pension and gratuity, of family pension and death-cum-retirement gratuity in respect of government servants dying while in service are also as per the rules laid down in the CCS pension rules. In addition, the rules also deal with sanction of family pension and residuary gratuity in respect of deceased pensioners, commutation of pension etc. There are 89 pension rules which are sub-divided into 252 sub-rules. Over and above these, there are 338 Govt. of India decisions in the form of Office Memoranda and Circulars which are in the form of case laws or new rules. These decisions have been announced from time to time taking into account the problems and hardships caused to a section of employees, by the main rules. The rules, sub-rules and govt. decisions for Commutation Rules are 34,73 and 23 whereas in the case of Extraordinary Pension Rules the numbers are 13,29 and 23 respectively.

We are at present writing the section on Qualifying Service as a logic program. This is the largest of the sections and relatively more complex in the formulation of the statutory laws. It contains 20 rules, 45 sub-rules and 95 Govt. of India decisions as case laws and/or new rules.

The main emphasis of this paper is in the use of temporal reasoning in the laws under consideration, the separation of the logic part of the program from the user data interface, the interaction of the user through forms with the knowledge base, the interplay of the propositional logic analysis of the rules with the entity-relationship analysis for the determination of predicates, parameters and conditions.

2. Logic Program Implementation Methodology

The program was first implemented in PROLOG without any considerations of logic and data interface problems. This mixing of user interface with knowledge representation led to complications in data management and logic management. At times for certain data values one could not be sure of the results of the program. It was difficult to ensure the correctness of the logic with data capture from the user and data manipulation interspersed all over the program.

The problem was then analysed de novo with Logic first. During the propositional logic analysis of the rules a strict discipline was maintained in keeping away from PROLOG and implementation problems. The idea was to identify appropriate propositional logic predicates. Soon it was discovered, that while one could identify the predicates to a reasonable degree of accuracy, the same was not true of conditions and parameters. In fact the choice of parameters seemed to affect clarity in so far as conditions were concerned.

We then tried the entity-relationship analysis of the variables involved to identify simple relations in the form of tables. The interplay of entity-relationship analysis with the propositional logic analysis led to the identification of minimum predicates with appropriate conditions and parameters. The logic of the problem could thus be completed. This is elaborated in the next section.

The analysis of the problem thus far was carried out without any worry of user interface with respect to data input. The program could be tested by supplying the required data as prolog facts. The user data interface (UDI) was designed separately so as to keep the logic of rules totally independent. A similar approach has been followed in the work on import export policy as well. It is proposed to develop a general shell as a user data interface for this class of problems.

The UDI captures the data at the start of the session. Simple forms have been designed for this purpose, which are similar to the manual forms being used by the offices. Necessary details concerning qualifying service, emoluments etc. are obtained through the filling of these soft forms.

Consultation with the knowledge base or the rules base begins at this stage and the program returns the results of consultation.

Finally, it is proposed to add a simple Explanation Module to the program. This will refer back to the logic program to pick up the applicable clauses and subclauses for a given situation and provide structured English of the clause as an explanation. We also propose to keep the English text of the corresponding clause verbatim from the book of rules. A prolog subprogram containing rule numbers and titles forms the simplest explanation module. There is thus a three stage approach to the automated reasoning system.

It may be noted here that unlike APES, the data is not being captured from the user interactively. The data is captured at the initial stage and then consulting with the logic begins. Thus this approach is an alternative to APES architecture. It is a general architecture applicable to a large body of problems. It is being used in the Import Policy problem referred to above. The main body of rules is a clean logic program free from user/data interaction. UDI, Logic and Explanation modules are three independent but interacting subprograms of the logic program.

3. Analysis of the Logic Problem

The rules on qualifying service relate to commencement of qualifying service, conditions subject to which service qualifies etc. The pension rules clearly specify how to treat the time spent on probation, whether pre-retirement civil service in the case of re-employed government servants counts and if so, under what conditions. Like this all aspects of service (leave, suspension, removal, reinstatement, resignation etc.) are covered in the section on qualifying service. It is obvious that the basic idea is to establish the initial date from which service should be counted in a given case. The uninterrupted service time period is expected as a result of application of this set of rules. Thus the concept of time is very crucial to this problem. The first-order logic makes it posible to take care of time explicitly thereby making temporal reasoning practical in a real life knowledge and inferencing problem.

As with a problem of this kind, we took the rules as laid down in the book in sequence and analysed with respect to logic. The entity relationship analysis was carried out subsequently which showed that the formulation of predicates was not entirely correct. This is best illustrated through an example. Rule 13 on qualifying service is as follows :

13. " Subject to the provisions of these rules, qualifying service of a Government servant shall commence from the date he takes charge of the post to which he is first

appointed either substantively or in an officiating or temporary capacity :

Provided that officiating or temporary service is followed without interruption by substantive appointment in the same or another service or post:

Provided further that -

(a) in the case of a Government servant in a Group 'D' service or post who held a lien or a suspended lien on a permanent pensionable post prior to the 17th April, 1950, service rendered before attaining the age of sixteen years shall not count for any purpose, and

(b) in the case of a Government servant not covered by clause (a), service rendered before attaining the age of eighteen years shall not count, except for compensation gratuity."

Predicates formulated for "commencement of qualifying service" included :

```
start-of-service (rule-no,Post,T1)
end-of-service(rule-no,Post,T2)
```

The entity relationship analysis and the need for reapplicability of predicates to later rules, however, led us to replace the "end-of-service(rule-no,Post,T)" predicate by "follows (Post, Post1)"

This was due to the fact that end of one service means start of another. With the former predicate we are storing redundant information and have to use 'start-of-service' after 'every end-of-service'. The 'follows' predicate, however, makes explicit the fact that the time of occurrence of the post corresponding to the second parameter is immediately after the post represented by the first parameter.

Direct representation of the text of rule 13 with respect to time using propositional logic analysis gave a representation such as

```
qual-serv(13,T1,T2) :-

(type(Post,officiating);type(Post,temporary) ),
start-of-service(T1),
end-of-service(T2).
```

However, with entity relationship analysis we handled the minimum age limitation in the predicate through the formulation given below :

```
real-start(13,Post,T1,T2) : -

service (Post,T1,T2),
start-qual-service(Post,T0),
(T0<T1, T=T1) or (T0>T1, T=T0)
```

The predicate start-qual-serv(13,Post,T) initially incorporating information regarding actual start of service so as not to count service before minimum age was also suitably amended as follows, owing to formulation of the generally applicable predicate real-start(13,Post,T) as defined previously. The entire rule 13 is formulated as follows :

```
start-qual-serv (13,Post,T) :-
     (type(Post,officiating);/*or*/type(Post,temporary),
     follows (Post,  Post1), type (Post1, substantive),
     service (Post, T1,T2),
     (exception-a(Post,Age,T1),age-check(Age,16,T,T1);/*or*/
     exception-b(Post, Age,T1),age-check (Age,18,T,T1) ).

exception-a(Post,A,T1) :-
     group(Post,D),lien(Post,17-apr-1950)
     service (Post,T1,T2),age (T1,A).

exception-b(Post,A,T1) :-

     not  exception-a(Post,Age,T1),
     service(Post,T1,T2),age(T1,Age).

age-check(Age,Threshold-age,T,T1) :-

     (  (Age < Threshold-age,T is T1 + Threshold-age - Age);
     (Age>=Threshold-age,T = T1)
```

The entity relating to 'Post' of an employee was initially to be a predicate allowing access to details such as group (A,B,C,D), organisation, type (apprentice, probationary, etc.). It was subsequently decided on the basis of simplicity and usability to make Post a structured term with all these attributes and have three predicates to access the components of this term given the post as a parameter. These three predicates would then be group, type, organisation, each taking as input the Post and giving as output the appropriate result. The structured term Post has been defined as

```
Post (Group, Title, Type, Organisation, Pay-scale, T1,T2)
```

We will consider one more rule and its formulation in Prolog before we leave this subject to discuss the more

specific and interesting aspects of CCS Pension Rules. Rule 14.1 alongwith its Logic is given below :

14.1 "The service of a Government servant shall not qualify unless his duties and pay are regulated by the Government, or under conditions determined by the Government."

```
qual-serv (14.1,Post,T1,T2) :-

     (regulated-by-govt   (Post)  ;/*or*/conditions-of-
     post (Post) ),
     (paid (Post,T1,T2,consolidated-fund);/*or*/
      paid (Post,T1,T2,local-fund) ),
      not(non-pensionable    (Post,T1,T2),not     qual-
      serV(_,Post,T1,T2) ),
      real-start (Post,T1).
```

There are several pension rules which are in the form of negative conclusions, which is like any other piece of legislation. Logic programs are known to represent knowledge in the form of implications

A if B_i and ...B_n , $n \geq o$

Negative conclusions have been shown to be represented as implications by adding extra conditions or transforming some of the existing conditions (4). A negated condition is deemed to hold if the corresponding positive condition can be shown to fail to hold - this is negation by failure (NBF). Negative statements are handled through NBF in the condition. We will examine some of the rules with negative conclusions.

16. "Service as an apprentive **shall not qualify**, except in the case of S.A.S. apprentice in the Indian Audit and Accounts Department or the Defence Accounts Department"

This statement is handled through suitable transformation of the conditions. The equivalent statement is, "service as an apprentice shall qualify, only in the case of S.A.S. apprentice in the Indian Audit and Accounts Department or the Defence Accounts Department. After the transformation this becomes

```
qual-serv (16,Post,T1,T2) :-

     type    (Post,apprentice),(organisation     (Post,
     Indian-audit-and-account);organisation
     (Post,defence-accounts) )
```

Another rule with a negative conclusion is rule

25.2 "The period of interruption in service between date of dismissal, removal or compulsory retirement, as the case may be, and the date of reinstatement, and the period of suspension, if any, shall not count as qualifying service unless regularised as duty or leave by a specific order of the authority which passed the reinstatement."

This can also be transformed to fit in logic programming. The equivalent statement is :

"The period of interruption in service between the date of dismissal, removal or compulsory retirement, as the case may be, and the date of reinstatment, and the period of suspension, if any, shall count as qualifying service if regularised as duty or leave by a specific order of the authority which passed the order of reinstatement." This gets easily translated into
predicate logic as :

```
qual-serv (25.2,Post,T1, T2) :-
     date-of-dismissal (Post,T1),
     date-of-reinstatement (Post,T2)
     suspension-period-regularised (Post,T1,T2)
```

We may also note here that rule 14.1 above is also an example of a negative conclusion and it has been represented through appropriate transformation of the conditions. We can safely conclude that the treatment of negative conclusions in logic programs for representing legislation as pioneered by Kowalski and others (4) is found to be adequate for this problem. We have had similar experience while representing the Indian import policy as a logic program.

We will now briefly consider the examples of those rules which exhibit dependence on time and/or result in computation of elapsed time as a result of the occurrence of certain events as laid down in the rules. This will illustrate the power of first-order logic in temporal reasoning.

The entire rule 13 as explained earlier in the context of the formulation of predicates is an example of temporal reasoning. The date of commencement of service is to be counted from the date of joining subject to an arbitrary date 17 April 1950 and the condition of whether the employee was under 18 or 16 years age depending upon his category. The age of the employee can be explicitly handled and checked against 18 or 16 years with respect to the date 17 April 1950 and a date arrived at unambiguously from where the service is to be counted. This rule has already been shown in its proper formulation in this section.

Similarly, the predicate 'follows (Post 1, Post 2)' also captures the movement of time in conjunction with other predicates. The rules applicable to an employee who works in a post Post1 from time T1 to T2 and in any break can be well represented through follows (Post1, Post2) :-

service (Post1,T1,T2),service (Post2,T2,T3)

4. User Data Interface

The data capture from the user is completed before consultation begins with the expert system. This is in contrast to the APES architecture (5) where data is captured interactively as consultation proceeds. In standard expert system shells also the data is obtained interactively from the user while the rules are tested by the expert-system.

The UDI has been developed based on forms approach which tries to present the users with forms on the screen. The entire information that needs to be collected from the user for computing pension, has been divided into a number of forms which are displayed on demand through a menu. The information relates to employee's personal details, retirement details, military service etc. The manual form has been categorised into the following screens which are shown as options to choose from a menu. The screens which must necessarily be filled by the user are shown with an asterisk.

1. Personal Details *
2. Retirment Details *
3. Military Service
4. Autonomous Organisation Service
5. Qualifying Service *
6. Govt. Dues and NOC action *
7. Commutation option

The requisite information is thus supplied by the user through these forms. After consultation with the Pension Rules knowledge base appropriate results are displayed on the screen.

5. Explanation Subsystem

The user can ask for explanation in which case the Explanation Subsystem is consulted by the expert system. The explanation part reconsults the knowledge base for the rules applicable to the case under consideration which have been saved after the initial consultation.

For the present only a simplified treatment of explanation is envisaged. The rule numbers alongwith their english text are stored which can be reproduced as part of explanation. The explanation subsystem can also interact with the knowledge base in which case structured text of the

rule can be presented which will be based on the way the rule has been written in propositional logic.

This portion of the expert system is still in preliminary stages.

6. Conclusion

We have presented the development of the CCS Pension Rules as a logic proram. The formulation of the rules in logic has clearly shown the need for interaction of propositional logic analysis with entity relationship analysis. The three stage approach of data capture through a UDI, knowledge base in the form of rules represented in logic, and explanation subsystem has been seen to be extremely efficient and useful from the viewpoint of practical development. The logic part of rules and the user interaction can be kept separately, which makes it easier to maintain the knowledge base. The UDI with its provision to capture the entire data from the user through screens at the initial stage has been shown to be a general architecture which is an alternative to APES. We hope this formulation will help us to show that the subsequent legislation can be framed without any ambiguities and in simple English.

REFERENCES

1. Sergot MT, Sadri F, Kowalski RA, Krivaczek F, Hammond P and Cory H T (1986) ; 'The British' Nationality Act as a Logic Program" CACM Vol 29 No. 5, pp 370-386

2. Bench-Capon J.J.M., Robinson G.O., Routen T.W., Sergot M.J. (May, 1989),"Large Scale Applications in Law : A formalism for Supplementary Benefit Legislation", Proceedings of the First International Conference on AI and Law, Boston, pp 190-198.

3. Bajaj K.K., Dubash R.K., Kamble A.S., Kowalski R A, Murthy B K, Rajgopalan D. (September 1989) : "Indian Import Policy and Procedures as a Logic Program" KBCS Nodal Centre, Department of Electronics, Govt. of India, New Delhi

4. Kowalski R.A. (June 1989) : "The treatment of negation in logic programs for representing legislation" Second International Conference on AI and law Vancouver, Canada

5. Hammond P., Sergot M.J. (1984), APES Reference Manual. Logic Based Systems Ltd., Richmond, Surrey, England, 1984