INDIAN CENTRAL CIVIL SERVICE PENSION RULES: A CASE STUDY IN LOGIC PROGRAMMING APPLIED TO REGULATIONS

M.J. Sergot[1] A.S. Kamble, K.K. Bajaj

KBCS Nodal Centre
Department of Electronics
Government of India
New Delhi

[1]Department of Computing
Imperial College of Science, Technology and Medicine
London SW7 2BZ

1. INTRODUCTION

Logic programming has been applied to a variety of examples in law and rules and regulations of different kinds. The best known examples are the representations of the British Nationality Act 1981 [Sergot et al. 1986] and United Kingdom social security legislation [Bench-Capon et al. 1987] constructed at Imperial College, London. The paper [Sergot 1988] and the survey [Sergot 1990] contain references to other applications. Essentially, these applications construct a legal analysis program by representing some fragment of legislation in logic and then executing the representation, either as a stand-alone program or as a component of some bigger system.

This paper describes an application of the same basic techniques to the Pension Rules of the Indian Central Civil Service (CCS). Employees are entitled to a pension and several other kinds of payments on retirement (or death), depending on various factors of which the length of (qualifying) service and rates of pay are the most important. The program calculates an employee's qualifying service and pension entitlements according to the rules in the standard reference book. An earlier paper [Bajaj et al. 1989a] presents a very preliminary version of this application.

The work was conducted at the KBCS Nodal Centre at the Department of Electronics (DOE), Government of India, as part of a much larger project on Knowledge Based Computer Systems (KBCS) sponsored by the Government of India and the United Nations Development Programme. Within this project the DOE centre is developing systems to assist with the administration and preparation of legislation.

At this stage the main objective of the DOE centre is to produce demonstration systems for distribution to potential users within the Government and elsewhere. The application described here is one of two that have been developed so far; the other concerns Indian import-export policy and regulations [Bajaj et al. 1989b]. The DOE centre has also embarked on a third application, dealing with case law in Indian Environmental Law, and water pollution cases more specifically.

Both demonstrator systems were developed on Apple Macintosh II computers using the MacProlog system [see references] and were then ported to IBMpc-style computers for wider distribution.

The CCS Pension Rules

The CCS Pension Rules determine the type and amount of pension and other retirement benefits for almost all Government employees in India. They also regulate how the pension scheme is administered. They are published in a standard reference book containing 89 rules, 252 sub-rules, and supplementary material in the form of Government of India Decisions (GIDs). GIDs are Office Memoranda and Circulars dealing with detailed procedures, or introducing expansions and clarifications to rectify problems that have come to light. The edition of the reference book we used contains 338 GIDs. There is also a shorter explanatory book *Know Your Retirement Benefits* which we will refer to from time to time. Most employees would consult this because it omits complicating detail and administrative procedures and gives a clearer overview of the scheme.

Pension entitlement depends essentially on qualifying service, which is roughly the period of time for which an employee has worked without interruption in Government service. Once qualifying service is determined calculation of pension entitlement is straightforward, requiring only some additional data on pay ('emoluments') and the types of post held. Chapter III of the CCS Pension Rules deals with qualifying service. It is the largest of the chapters, containing 20 rules, 45 sub-rules and 95 GIDs. It is also the most complex. This paper describes only the component for determining qualifying service. The parts dealing with calculation of pensions are very straightforward in comparison.

Aims and structure of the paper

The main aim of this paper is to describe the program and the techniques used in its construction, but there are some features we wish to emphasize more than others.

The bulk of the paper comprises a description of the CCS Pension Rules for 'qualifying service' (section 2), the general design of the system (sections 3 and 4), and extracts from the representation itself (section 5). Because of the nature of the Rules and what the program computes, much of this discussion concerns the representation of periods of time. This is not the feature we wish to emphasize here. There is nothing specifically 'legal' about this aspect of the representation, so we present only what appeared to be the simplest treatment of time with a few remarks about possible alternatives. The extracts in section 5 are fairly detailed, but this is only to give an impression of what the Rules are like.

The features we wish to emphasize are the nature of the program, and the nature of the Pension Rules.

Stamper took as the point of departure for his LEGOL project [Stamper 1980] the observation that many typical data processing systems are introduced into an organisation to administer rules and regulations, and he advocated that this should be made explicit in the analysis and implementation of such computer systems. Following Stamper, Sergot [1988] argues that there is no *essential* difference between something like a payroll system and a legal analysis program that attempts to determine whether the legal concept of 'taxpayer' applies to the facts of some given case, except that the programming techniques employed in the two cases are usually different. The CCS Pension Rules program provides a simple illustration in support of this argument. Though it can be regarded as a legal analysis program in that it gives precise definitions of quasi-legal concepts like 'qualifying service' expressed in logic, it can also be regarded as a program to automate the calculation of employees' pension entitlements, which is a fairly typical kind of data processing application.

Second, legislative provisions and regulations vary widely in their character. Moreover, the same regulation can be read and represented in different ways depending on what the representation is for. The CCS Pension Rules provide some good examples. Some of the rules, which at first sight contribute to the definition of 'qualifying service', are better regarded as constraints on how the pension scheme operates and is administered. We give some specific examples in section 5 and indicate in section 6 how the representation would differ if we wished to construct a program supporting the administrative aspects of the Pension Rules.

We should have liked to document here the process by which we arrived at the representation. We could not for lack of space. We present only the end result with occasional comments in the text and some remarks about alternative formulations in the concluding section.

2. THE CCS PENSION RULES AND QUALIFYING SERVICE

The basic idea of qualifying service is simple enough. A person works in the Central Civil Service without interruption in a variety of posts. The total period of time spent in these posts is essentially the period of qualifying service, but there may be isolated periods of time that have to be subtracted, such as periods of suspension or training, time spent on extraordinary leave, and so on. It is also possible that some or all of this service must be discounted altogether, because certain circumstances, such as resignation or dismissal, entail forfeiture of past service.

Ultimately, we are not as interested in the *period* of qualifying service (such as 25 January 1955 to 13 May 1986) as in the *amount* of qualifying service, which is expressed as the number of completed six-monthly units (here 62 of them). The *period* of qualifying service, with its start and end dates, is just a step in calculating the amount of qualifying service – but it is the main step and the period itself cannot be discarded because the actual dates are required later, for example to calculate the amount of pension paid.

Imprecision

We do not wish to dwell on how many representational problems are caused by imprecision or other flaws in the drafting, except to make this remark.

In common with many other examples of legislation and regulations, especially those that refer to periods of time, the Pension Rules are imprecise and very casual about many of the key concepts. They are certainly not precise enough to be formulated directly as an executable program, and they are arguably not precise enough to be applied by a human agent either.

The main problem, apart from the way the Pension Rules are structured, is that there are clearly some underlying concepts that are key to understanding and applying the rules but which are not defined or even mentioned directly. For example, the period of qualifying service seems to be a key idea, but this concept is not referred to explicitly. Indeed, the very phrase 'qualifying service' as used in the Pension Rules often does not refer to the qualifying service that is being defined, but to the different concept 'service that qualifies for pension' or even 'service that would qualify for pension but for certain other conditions'.

Before a representation of the regulations can begin, in any representational language (including English), it is necessary to establish a more precise vocabulary of underlying concepts.

Of course it has been remarked before that legislators and drafters of regulations often use the same term to refer to quite different things, and that this can cause severe difficulties in interpreting what has been written. We have made the remark again because in this application the wording and structure of the Rules impeded representation to such an extent that it was impossible to make much progress without devising our own terminology.

Qualifying posts and disqualified periods

We need some way of referring to 'service that qualifies for pension' and distinguishing this from 'qualifying service' as defined by the Rules.

Time spent in posts outside the Central Civil Service can sometimes count towards qualifying service. Examples are service in one of the state Governments followed by transfer to central Government, and certain types of military service. Conversely, some posts within the Central Civil Service, such as some probationary appointments and most apprenticeships, do not count towards pension.

In describing and representing the Pension Rules, we speak therefore of qualifying 'posts'. The term 'post' refers to an appointment with a specific title and a specific grade at a specific organisation within Government (such as 'Senior Engineer, grade A, Department of Electronics'). An appointment in a different organisation, or with a different title or a different grade, is a different 'post'. A 'qualifying post' is one which the Pension Rules say contributes to qualifying service; we thus avoid 'service that qualifies for pension' and speak instead of 'time spent in qualifying posts'.

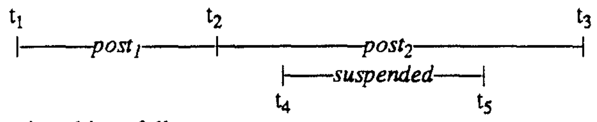
Some periods of time in qualifying posts do not contribute to qualifying service. Periods of suspension (rule 23) fall into this category, and so do some condoned interruptions in service (rule 28). These are examples of periods of time that have to be discounted when calculating the amount of qualifying service. We call them 'disqualified periods'.

Note that Civil Service appointments which do not contribute to qualifying service, such as the probationary appointments and apprenticeships mentioned above, can be viewed in two ways: either as posts which do not qualify, or as posts which do qualify but whose periods of time are 'disqualified periods' that are discounted when calculating the amount of qualifying service. The second view seems somewhat contrived but corresponds roughly to what is said in the explanatory booklet. It also has some other benefits to do with explanations which we point out in section 5.

Ignoring 'forfeiture' for the moment, 'qualified posts' and 'disqualified periods' give a simple conceptual framework in which to express the definition of 'qualifying service'. There are some additional complications, such as periods of time that count only fractionally towards qualifying service, but these are minor complications because they just require elaborating the formula for calculating the amount of qualifying service to take a weighting factor into account. Similarly, there are circumstances in which an employee may be awarded an additional discretionary amount of qualifying service (rules 29 and 30). Again this is a minor complication because it does not affect the *period* of qualifying service but only the formula for calculating the *amount* once the period is determined. We omit this level of detail when describing our representation in this paper.

Example

A person holds two qualifying posts in his or her Civil Service career, including one period of suspension (say) which does not count towards qualifying service.



We view this as follows:

- the total period of time spent in qualifying posts ('total service of the Government servant' in the explanatory booklet) is
  t_1–t_3
- from this we must discount the disqualified period t_4–t_5;
- so the *period* of qualifying service consists of two time intervals
  t_1–t_4 and t_5–t_3
- the *amount* of qualifying service (*qs*) is given by
  qs = *six-month-units-in*(t_1–t_4 "+" t_5–t_3)
  There may also be a discretionary amount to be added to *qs* but this does not affect the main calculation. Note that, because of rounding, this formula may or may not yield the same answer as
  qs = *six-month-units-in*(t_1–t_4) + *six-month-units-in*(t_5–t_3)
  or
  qs = *six-month-units-in*(t_1–t_3 "–" t_4–t_5).
  The Pension Rules and the explanatory brochure do not address the details of rounding, but since our program calculates the relevant periods, adjustments to the *qs* formula can be accommodated very easily.

Qualifying Service: forfeiture

A complication in formulating the definition of qualifying service is that dismissal or removal or resignation from Government service (and certain other circumstances) entails forfeiture of past service. Indeed, Rules 27 and 28 say that, with some exceptions, any interruption in the service of a Government service forfeits past service.

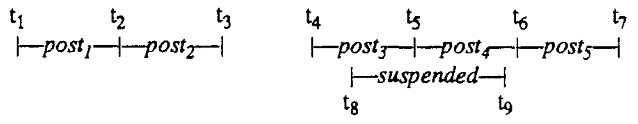
There are two main reasons why forfeiture of past service is a complication.

- First, without the possibility of forfeiture, determining whether a given post contributes to qualifying service simply requires looking to see what kind of post it is. With the possibility of forfeiture this is not enough. Service which would otherwise qualify may have been forfeit later, so it becomes necessary to examine also the rest of the service record.
- Second, there are now two different kinds of interruptions in service. There are interruptions which are just disqualified periods that have to be subtracted from the total period of time in qualifying posts. And there are interruptions which are more serious than this, because they forfeit all past service too. The rules for distinguishing between these interruptions are dispersed throughout the text and are sometimes implicit. 'Condoned interruptions' do not forfeit past service - but some do and some do not have to be subtracted when calculating the amount of qualifying service.

The Pension Rules often refer to 'qualifying service' where a phrase like 'service that qualifies if it is not forfeit later' would be more accurate. Again, 'qualifying post' eliminates the need for circumlocutions like this: a post is a 'qualifying post' or it is not; time spent in a qualifying post may count towards qualifying service, or it may not.

Example

Suppose a person holds two qualifying posts, followed by a forfeiting interruption, followed by three more qualifying posts after re-employment. Suppose also that the second series of posts contains a disqualified period of suspension. (It does not matter whether this example is realistic. It is used for illustration only.)



In our terminology:

- the total period of time spent in qualifying posts consists of two separate time intervals t_1–t_3 and t_4–t_7
- the interruption t_3–t_4 forfeits all past service, leaving t_4–t_7;
- from this we must discount the disqualified period t_8–t_9;
- so the period of qualifying service consists of t_4–t_8 and t_9–t_7;
- the amount of qualifying service (*qs*) is
  qs = *six-month-units-in*(t_4–t_8 "+" t_9–t_7)
  or qs = *six-month-units-in*(t_4–t_8) + *six-month-units-in*(t_9–t_7)
  or qs = *six-month-units-in*(t_4–t_7 "–" t_8–t_9).
  depending on the details of rounding.

We stress that this is our own terminology. These concepts are not referred to explicitly in the Pension Rules, although they do seem to correspond quite closely to the account given in the explanatory booklet. Within this conceptual framework, there are many possible formalisations of the individual Rules and the manipulation of time periods. The formalisation we used in the demonstrator system is sketched in section 5.

3. DESIGN CONSIDERATIONS

We want a system that calculates an employee's pension entitlements (qualifying service) according to what is written in the Pension Rules.

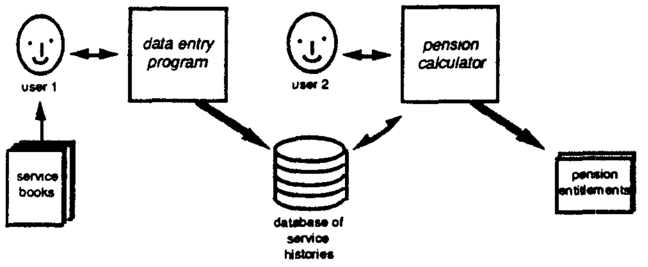
It is immediately obvious that a certain amount of data - details of the posts held during the employee's Civil Service career and other relevant periods of time - will have to be input at some stage. It is possible to arrange that all data will be requested automatically by the program as it calculates, but to leave all data entry until then is unrealistic in this application. We are going to have to input at least the basic data regarding the service history, and it would be natural if this takes place ahead of time, before calculation of the pension entitlements begins. In any case, in an operational setting, it is very likely that many of these data would already be available on the personnel databases that Government departments maintain, and the application should reflect this if it is to be a convincing demonstrator.

Every employee in the Indian Civil Service has a 'service book', which is a (paper) record of the complete career history, including details of appointments, transfers, promotions, periods of sickness, leave, deputation to other organisations, and so on. It is the information in the service book from which pension entitlements are calculated. Several of the GIDs in the CCS Pension Rules stress the importance of making proper entries in it.

The demonstrator system we have implemented has two separate components:

- the *pension calculator*, which is a set of rules defining qualifying service and pension entitlements in terms of assertions about employees' service histories;
- a data entry program (in Prolog) for transcribing the information in the 'service book' into a set of assertions stored as a (Prolog) database.

The two programs serve separate functions, though in the demonstrator system they are normally resident in the machine at the same time.



In addition to the raw data about an employee's Civil Service posts and the other recorded periods of time, there are many further details which need to be input, such as whether a period of suspension was 'subsequently regularised as duty or leave', or whether a transfer to another post was 'by deputation only' or whether a temporary employee held a 'lien or a suspended lien on a permanent pensionable post'. In the demonstrator system these additional details are not collected by the data entry program but are referred to the user of the pension calculator itself ('user2' in the diagram). The rules defining pension entitlements are executed by a Prolog system which provides a 'Query-the-User' (QtU) mechanism [Sergot 1983]. The QtU mechanism automatically generates requests for additional data as the pension calculation requires them.

(Most of the applications of logic programming referred to in the introduction used the APES system [Hammond & Sergot 1983] which provides a general QtU mechanism as well as several other features. In this application, a very simple version of QtU is adequate and can be implemented in just a few lines of Prolog code.)

A note on methodology

The use of a QtU mechanism in this application might seem to be redundant. But eliminating it from the outset places too great a burden on the data entry program. Not only must the data entry program provide a convenient user interface, it must identify what data are relevant and whether some items are worth collecting at all, and it is always necessary to check that data are meaningful as well. In the worst case, the whole of the Pension Rule representation would migrate into the data entry program and be absorbed in the procedures that implement it.

Experience in previous applications has demonstrated the value of decomposing the development of an application into two separate phases. The objective of the first phase is to produce a representation of the required fragment of law. This representation should be executable, but the emphasis is on producing an accurate representation and not on operational and behavioural questions. These are adjusted later in a separate second phase. The main advantage of this decomposition is that, by and large, all legally sensitive issues concerning representation are addressed in the first phase, while only computational details are examined in the second. Since the first phase is critical, it is important not to confuse and complicate this by trying to address at the same time how the program should behave, what sort of questions it should ask, and in what order.

It might be desirable eventually to move some or all of the QtU interaction into the data entry program. But this can be done in a later development. (Part of it has been done already.) Note that no adjustment is required to the pension calculator and the rules defining pension entitlements. The QtU mechanism only generates requests for *missing* data; if an item of data has already been entered the QtU mechanism is not invoked.

A possible third component

Calculation of pension entitlements takes place when an employee is about to retire (or has recently deceased). In our system, the function of the data entry program (and the QtU mechanism) is to transcribe the information that is present in the service book at this stage.

It would be possible of course to add a third program to the system, which would allow an electronic version of the service book to be built up and maintained as the employee's Civil Service career progresses. This third component has not been built. Section 6 sketches what construction of such a program would involve by reference to some specific examples. It argues that the representation of the Pension Rules would have to be different and would require attention to some interesting technical questions. For the time being, we have concentrated on the much simpler problem of calculating pension entitlements from the information that is present in the service book at the time of retirement.

Treatments of temporal data

The representation of the rules defining pension entitlements, and to some extent the data entry program too, depend on how we choose to deal with temporal data in this application. Since we are not emphasizing the treatment of temporal data in this paper we make only some general remarks here.

Apart from special-purpose temporal logics, which are not particularly useful in this application, there are two basic schemes for dealing with temporal information: one is based on intervals or periods of time; the other is based on events, or the time points at which changes take place. In this application, both schemes can be used, or any combination of them.

For the demonstrator system we have chosen to formulate both data about service histories and the rules defining pension entitlements in terms of periods only. This seems to yield the simplest treatment, though not perhaps the most elegant. The simplicity of the treatment is important because we want to be able to explain to users how the calculations were obtained. Suggestions for other treatments of temporal data are mentioned in the concluding section of the paper.

4. THE DATA ENTRY PROGRAM

This section gives a brief description of the data entry program. It is included for the sake of completeness only, even though the time spent on construction of this program far outweighs the time spent on representation of the Pension Rules themselves.

There are two parts to the data entry program: input of data, and the checking of constraints to ensure that the data are meaningful.

Data about an employee's service history are input through a series of forms. This is also the interface for examining and editing the database of service histories. It was the implementation of this part which took most of the programming effort, because of the amount of low-level programming required. (MacProlog provides a set of routines for generating 'dialogs' and menus at a comparatively high-level, but much time-consuming programming is still necessary to set up the required forms. Moving the whole demonstrator system to IBM-style personal computers is almost exclusively a matter of re-implementing the forms interface.) All of this programming effort could have been eliminated by arranging for the data to be input using any standard database or spreadsheet package, except that so many items of data have to be entered and so many constraints have to be checked that implementing a special-purpose data entry program in Prolog became more attractive.

The constraint checking techniques employed have some technical interest but cannot be presented here for lack of space. It is enough to say that the data entry program ensures that descriptions of posts and other recorded periods of time together make up a coherent service history. In particular, gaps in the service history, which can affect pension entitlements, could be due to mistakes in data entry or to genuine interruptions in service. The difference can be detected because genuine interruptions only occur in certain well-defined circumstances.

5. DEFINITION AND CALCULATION OF QUALIFYING SERVICE

As already mentioned, the essential problem in calculating pension entitlements is to determine the amount (or the period) of qualifying service. Calculation of the various pension amounts follows immediately from this and we omit the details. The demonstrator system also provides an 'explanation' component which justifies how the calculation was obtained. Again, its implementation is straightforward and we do not present the details.

Chapter III of the CCS Pension Rules deals with qualifying service. Every one of the 20 main rules, with its supplementary GIDs, raises interesting representational questions. We cannot examine all of the options in detail. We describe only how our formulation deals with the temporal data (periods of time) and sketch how the individual Rules are represented within this framework. The Rules for qualifying service can be grouped naturally into several categories, roughly in the order in which they appear in the text. We give representative samples of each group, and make further comments about some of them in section 6. In order to give an impression of what the representation involves (and why the program performs a useful computation) these extracts are fairly detailed and this section of the paper is comparatively long.

Form of the input data

Assume that the basic service history of each employee is described as a set of assertions of the following form:

employee(*Emp_id, Date_of_birth*)
post_held(*Emp_id, Post*)
recorded_period(*Emp_id, Period*)

(This is not the exact form of the data constructed by the data entry program but there is an interface which converts what is constructed into this form.)

Here *Emp_id* is some identifier for the employee. *Post* in the `post_held` assertions is a data structure (a Prolog term) giving details of a single Government post held by the employee, including start and end dates, the organisation, position held, details of the type of post, and so on. We explain in a moment how the formulation of the qualifying service Rules can be made independent of the details of this data structure. *Period* in the `recorded_period` assertions is a data structure for other relevant periods of time, such as periods of suspension, leave, sickness etc., again with start and end dates and other details as provided through the data entry program.

In our terminology, each of the Rules roughly speaking gives conditions that determine whether time spent in a given post counts towards qualifying service, which other periods of time are disqualified and hence to be subtracted, and what kind of interruptions in service entail forfeiture of past service.

A simple example of a Rule

Rule 13 is the first in the section on qualifying service and the one that applies to almost every post held by every Civil Service employee. With some exceptions, it says simply that qualifying service commences from the first substantive appointment. No Rules specify when qualifying service ends, this being left to the common-sense of the reader. In our conceptualisation, we need to determine whether time spent in a 'post' is to be regarded as service that qualifies, and Rule 13 is formulated thus:

```
qualifying_period('13', Emp_id, Period) :-
   post_held(Emp_id, Post),
   category(Post, substantive),
   type(Post, central),
   % 'central' means rules 14(1) & 14(2) apply
   time_period_of(Post, Period).
```

(We use Prolog syntax throughout. Strings beginning with an upper case character, and the underscore _ , are variables.)

The choice of the predicate `qualifying_period` could perhaps be improved. Its intended reading is 'time spent in service that qualifies' and not the 'period of qualifying service' (which is what we want to compute). The first argument '13' refers to the Rule and is not significant - it is only used by the explanation module.

Note that the representation of Rule 13 does not refer directly to the internal structure of `Post` but uses predicates like `category`, `type`, and `time_period_of` to select items from it. The use of these 'selector' predicates is a standard programming technique to make the formulation of rules independent of changes in data structures.

Suppose that the `Post` data structure is a Prolog term of the form:

post (***Organisation, Type, Position, Grade, Category, Start, End***)

'Selector' predicates can be defined straightforwardly:

```
type(post(_,Type,_,_,_,_), Type).
category(post(_,_,_,_,Category,_,_), Category).
time_period_of(post(_,_,_,_,_,St,End), St-End).
```

Additional 'selector' predicates can be defined as they are required. If the post data structure has to be modified later, for example to include other items of data that turn out to be necessary, then no modification is required to the rules, but only to the definition of the 'selector' predicates. (The post data structure used in the application is more complicated than the one illustrated but there is no need for us to explain all the details, for the reasons just given).

Note also the comment in the rule shown above. Rules 14(1) and 14(2) of the Pension Rules give conditions for what counts as 'central Government'. For the pension calculator we assume that these conditions are brought to the user's attention during transcription of the service book, and that only posts satisfying them have been entered as type 'central'.

Exceptions

One of the exceptions to Rule 13 is that 'service rendered before attaining the age of eighteen years' does not count. Since disqualified periods of time will be subtracted when calculating the period of qualifying service, they effectively give us a way of expressing negation, and thus a way of expressing exceptions too. Hence:

```
disqualified_period('13', Emp_id, Discounted) :-
   post_held(Emp_id, Post),
   attains_age(Emp_id, 18, Date),
   time_period_of(Post, Period),
   period_before_date(Period,Date,Discounted).
```

(This is only an approximation of the actual rule because the minimum age is dependent on the type of post and other details, but the simplified version is sufficient for present purposes.)

`attains_age` and `period_before_date` are just auxiliary procedures for manipulating dates.

Most of the conditions in this rule are redundant. It does not matter whether the employee did or did not hold a Government post at the time of the eighteenth birthday because no period of time before the eighteenth birthday qualifies for pension, for Government employees or for anybody else. Viewed computationally, it is not *durations* of disqualified periods that are 'subtracted', but the periods themselves. The period up to the eighteenth birthday will be subtracted, but the subtraction will have no net effect if there is no qualifying period to subtract it from. We include the redundant conditions in the rule for the sake of explanations. Without the redundant conditions, an employee who did not begin Government service until he was twenty-five years old would be told that the period up to his eighteenth birthday does not qualify for pension. Though true, this is a vacuous and potentially confusing statement.

A more representative example

Rule 13 also allows officiating or temporary appointments to count towards qualifying service, but only if they are followed without interruption by substantive appointment of the right kind.

```
qualifying_period('13', Emp_id, Period) :-
   post_held(Emp_id, Temp),
   (category(Temp, officiating)
      ;           % 'or' in Prolog
   category(Temp, temporary)),
   type(Temp, central),
   next_post(Emp_id, Temp, Next),
   category(Next, substantive),
   type(Next, central),
   without_interruption(Temp,Next),
   time_period_of(Temp, Period).
```

Many of the rules refer to the next post in the service history. `next_post` just summarises the conditions so that they do not have to be written out every time.

Again, the rule as presented here is a simplified version. It deals with a single temporary 'post' instead of temporary 'service' - which we represent as sequence of temporary posts.

Temporary service qualifies *only if* it is followed by substantive appointment. Since disqualified periods give us a way of expressing negation, the 'only if' part of the Rule can be expressed

```
disqualified_period('13', Emp_id, Period) :-
   post_held(Emp_id, Temp),
   (category(Temp, officiating)
      ;           % 'or' in Prolog
   category(Temp, temporary)),
   type(Temp, central),
   not (
      next_post(Emp_id,Temp,Next),
      category(Next, substantive),
      type(Next, central),
      without_interruption(Temp,Next)),
   time_period_of(Temp, Period).
```

Computationally, this rule has no effect in the calculation of qualifying service, since a temporary post which does not satisfy the conditions of Rule 13 is not a qualifying post and its period of time will not contribute to qualifying service anyway. The negative 'only if' rule is useful for explanations. An employee will want to know why his period of temporary service has not been taken into account: explaining why this period is disqualified is easier than explaining why it fails to qualify.

The 'without interruption' condition in this rule also needs some comment since it appears in many rules and also affects the way that period of qualifying service is defined and computed.

'Without Interruption'

Our current representation of 'without interruption' is very simple and possibly inadequate.

We take it that $post_2$ follows $post_1$ without interruption if $post_2$ starts on the day $post_1$ ends, or (more usually) on the day after. But it is conceivable that $post_1$ ends on a Friday and $post_2$ starts on the following Monday, or that some public holiday intervenes, and that the two posts follow each other 'without interruption' nevertheless. This sort of detail could be accommodated easily enough if we implemented and referred to a calendar. We have not done so, leaving it to the data entry program to ensure that start dates and end dates of posts are entered by convention in such a way that this kind of trivial but irritating complication is eliminated.

A possibly more serious omission is that our 'without interruption' test does not take into account what the Pension Rules call 'condoned interruptions' in service. Arguably, but not certainly, two posts separated by a condoned interruption follow one another 'without interruption'.

We have not included a 'condoned interruption' condition in our 'without interruption' test because it seems to be superfluous. We assume that the data entry program detects unexplained gaps in the service history and asks its user to correct those that are genuine mistakes. It is possible that interruptions in service still remain. The significance of condoned interruptions lies in that *uncondoned* interruptions apparently forfeit past service. Yet when examined more closely, there seems to be no imaginable circumstance covered by the forfeiture rules which would not already be covered by some other recorded period in the service history. (The forfeiture rules are described later.) Our current representation of 'without interruption' relies on this analysis; in case it is mistaken, the 'without interruption' condition is defined separately so that it can be adjusted later if the need arises.

A rule requiring additional data

Sub-rule (3) of Rule 14 says that service in a state Government counts towards pension if it is followed by transfer to central Government by deputation only.

```
qualifying_period('14(3)', Emp_id, Period) :-
   post_held(Emp_id, State_post),
   type(State_post, state),
   next_post(Emp_id, State_post, Next),
   type(Next, central),
   transfer_by_deputation(Emp_id,State_post,Next),
   time_period_of(State_post, Period).
```

Again we present a simplified version of the actual rule, dealing only with a single state post rather than service in a series of such posts, and we have not shown the 'only if' `disqualified_period` half of this rule. It is also unclear from the wording whether Rule 14(3) requires the central post to be a substantive appointment but that is not the point of this example. The point is the `transfer_by_deputation` condition. In the current implementation all such conditions are handled by the QtU mechanism which refers them to the user of the pension calculator. It might be preferable eventually to move some of these questions into the data entry program. (The formulation of the rule itself will not change because `transfer_by_deputation` will just become another 'selector' predicate.) But at this stage we prefer not to complicate the data entry program by making it deal with this sort of detail.

Note incidentally that there is no 'without interruption' condition in Rule 14(3). This might be an oversight, or it might be implicit in transfer by deputation. (Just in case, we included the 'without interruption' condition in our version of the rule.)

Other Pension Rules specify which other service or posts qualify. They are more or less complicated but are represented in similar style.

Other recorded periods

Several Pension Rules deal exclusively with what we are calling 'other recorded periods', specifying conditions under which they are 'disqualified periods'. Rule 23 deals with suspension from service for disciplinary reasons and is a typical though comparatively simple example.

Rule 23 says (in our terminology) that a period of suspension is not a disqualified period if the Government servant is 'fully exonerated' or the suspension is 'held to be wholly unjustified'. There is no vagueness in these conditions because a GID specifies the procedure for deciding and recording such questions. Otherwise, the period is disqualified unless the competent authority expressly declares that it counts.

```
disqualified_period('23', Emp, Time_period) :-
   recorded_period(Emp, Period),
   type(Period, suspension),
   not exonerated_or_unjustified(Emp, Period),
   not expressly_declared_as_qualifying(Emp, Period),
   excluded_by_entry_in_service_book(Emp, Period),
   time_period_of(Period, Time_period).
```

Again, the conditions in this rule are dealt with by the QtU mechanism or as 'selector' predicates on the data structures stored using the data entry program.

The condition `excluded_by_entry_in_service_book` comes from GID(1) under Rule 23, which stresses the importance of making proper entries about suspensions in the service book. GID(1) finishes with: "Specific entries in this regard in the service book will be taken note of at the time of reckoning qualifying service. In the absence of any specific entry, period of suspension (sic) shall be taken as counting towards the qualifying service."

This simple treatment of GID(1) is made possible because of the assumption that the program is used to calculate pension entitlements at the time of retirement, when the service book is complete. GID(1) (and GID(2) dealing with detailed administrative procedures) would have to be represented differently if we were to construct the missing third component referred to earlier, the program that allows an electronic version of the service book to be maintained. We sketch what would be involved in constructing this third component in section 6.

Other recorded periods are dealt with in the same way as periods of suspension. In the simplest representation there are also recorded periods corresponding to 'condoned interruptions' that would forfeit past service if they were not 'condoned'. These are described next.

Forfeiture of past service

Five of the Pension Rules (24 to 28 inclusive) are devoted to forfeiture of past service. But for one minor exception which we mention at the end, it seems on closer examination that there are only two circumstances which entail forfeiture: dismissal or removal from service (Rule 24) and resignation (Rule 26). The other three Rules contain exceptions, expansions and (apparent) repetitions.

Both types of forfeiture have the same basic form: there is (what we call) a 'forfeiting event' (dismissal or resignation); there are exceptions (re-instatement after dismissal or withdrawn resignations); and there are conditions which determine whether a condoned interruption (dismissal–re-instatement or resignation–withdrawal) counts towards qualifying service or is another case of a 'disqualified period'.

There are many different ways of representing these forfeiture rules depending on how we imagine the circumstances are most naturally described.

For example, we could describe a dismissal–re-instatement episode as two separate posts:

|—*post*$_1$—| |—*post*$_2$—|
dismissal   re-instatement

This sort of description seemed unnatural to us, especially as *post*$_1$ and *post*$_2$ will normally be identical in almost every respect.

An alternative is to say that there was only one post, but it included a recorded period of time of type 'dismissal–re-instatement':

|————————*post*————————|
|—*dismissal–re-instatement*—|

We chose the second of these. Both views (and others) support the representation of forfeiture rules straightforwardly enough, but the second seemed to correspond more naturally to the description that would appear in a (complete) service book. There would be a record of the dismissal, followed by a statement that the employee was subsequently re-instated and resumed duty, and it seems reasonable to transcribe this information as another kind of 'recorded period'. (Of course it could be argued that neither of these alternative ways of transcribing the service book is adequate, and that a narrative account would be most natural. We make some comments along these lines in the concluding section of the paper.)

We illustrate the representation of forfeiture rules by showing how resignation is treated. Dismissal is almost identical but contains a few more conditions.

```
forf_event('26(1)',Emp,forf(resignation,Date)) :-
   post_held(Emp, Post),
   exit_reason(Post, resignation),
   end(Post, Date).
```

Here `exit_reason` and `end` are 'selector' predicates for the `Post` data structure. In the term `forf(resignation, Date)` only `Date` is significant for calculating qualifying service. The type of forfeiting event (here 'resignation') is recorded for the use of the explanation module. Note that there is no exception in this rule because a withdrawn resignation does not end a post in our chosen formulation.

The period between a withdrawn resignation and resumption of duty never counts towards qualifying service. So we have also

```
disqualified_period('26(6)', Emp, Time_period) :-
   recorded_period(Emp, Period),
   type(Period,withdrawn_resignation-resumption),
   time_period_of(Period, Time_period).
```

(We referred earlier to the existence of one other type of interruption that can entail forfeiture of past service. Although Rule 27 states that any interruption in service forfeits past service, every circumstance we can imagine, bar one, is covered by an exception, or resignation or dismissal. The remaining circumstance is participation in a strike. We have omitted strikes from our representation since some of what is said in the GIDs suggests that participation in a strike would only forfeit past service if followed by disciplinary action, suspension from duty, and dismissal, which we already cover. If we are wrong then a strike could be represented as another kind of 'recorded period' with a forfeiting event of type 'strike' occurring at its start.)

The period of qualifying service

We now have all the elements to define the period of qualifying service. For clarity we present first a simplified version that ignores forfeiture and then show how it can be adjusted to take forfeiture into account as well.

Ignoring forfeiture:

```
period_of_qualifying_service(Emp, Qual_service) :-
   findall(QP,
           qualifying_period(_,Emp,QP), QP_list),
   amalgamate_periods(QP_list, Qual),
   findall(DP,
           disqualified_period(_,Emp,DP), DP_list),
   amalgamate_periods(DP_list, Disqual),
   subtract_periods(Qual, Disqual, Qual_service).
```

Here `findall` is the Prolog primitive that finds all solutions to a goal: a call to `findall` (*Term,Goal,Solutions*) returns *Solutions* as the list of all bindings of *Term* corresponding to successful executions of *Goal*. (Some Prolog implementations do not provide `findall` directly but have `setof` or `bagof` which do roughly the same thing.)

Read procedurally, the first two conditions of the clause defining the period of qualifying service collect the list of all periods of time spent in qualifying posts (`QP_list`) and 'amalgamate' them into periods of continuous service (`Qual`). The next two conditions find the list of all disqualified periods of time (`DP_list`) and amalgamate them into continuous periods (`Disqual`). The last condition subtracts the disqualified periods from the periods spent in qualifying posts. The implementation of programs to amalgamate and then subtract periods of time is a straightforward exercise and we omit the details. (The amalgamation step could be eliminated, but it makes `subtract_periods` much easier to write. It also isolates occurrences of the problematic 'without interruption' condition.)

The purist might object to the claim that this representation provides an adequate definition of `period_of_qualifying_service` because the `findall` operator is extra-logical and certainly not first-order. The criticism is valid, but nothing in the formulation depends on the use of `findall`. It can be eliminated easily, at the expense of making the formulation more tedious to write and more difficult to read.

The adjustment to take forfeiture into account is very simple. We could find all forfeit periods and subtract these as well, but it is easier just to ignore all forfeit qualifying periods from the outset:

```
period_of_qualifying_service(Emp, Qual_service) :-
   findall(QP,
           (qualifying_period(_,Rule,QP),
            not forfeit(_,Emp,QP)),QP_list),
   amalgamate_periods(QP_list, Qual),
   findall(DP,
           disqualified_period(_,Emp,DP), DP_list),
   amalgamate_periods(DP_list, Disqual),
   subtract_periods(Qual, Disqual, Qual_service).
```

A period is forfeit if it is in the past of some 'forfeiting event':

```
forfeit(Rule,Emp_id,Period,forf(Type,Date)) :-
   forf_event(Rule,Emp_id,forf(Type,Date)),
   end(Period, End), End ≤ Date.
```

6. THE (MISSING) THIRD COMPONENT

Every Government employee's pension entitlement is calculated from the information that is present in the service book at retirement (or death). The pension calculator described in the previous section performs the calculation; the data entry program is just a means of transcribing what is in the service book so the calculation can be performed.

There are other programs that could be built to complement this pair of programs. An obvious additional component would be a program that maintains a version of the service book as some kind of database, allowing entries to be made as the employee's Government career progresses. We are thinking here not of a program that just keeps an electronic copy of what currently appears on paper, but a different administrative system where the paper service book is dispensed with altogether and *replaced* by a database. And we suppose that Pension Rules which currently refer to the service book would apply now to the electronic version.

This third program would also need to incorporate a representation of the Pension Rules, since they lay down administrative procedures and they impose constraints on what can be entered in the service book and when these entries can be made. But the representation of the Rules for this program would have to be different. What is done with the representation would have to be different too.

The requirements are most clearly illustrated by reference to a specific example. We use Rule 21 dealing with extraordinary leave. This is similar to Rule 23 for periods of suspension which we sketched in the previous section, but it applies more often and covers a wider range of circumstances.

Imagine the following scenario. An employee is granted leave (noted in the service book) but returns to duty a month later than he was supposed to. Rule 27(1) says that unauthorised absence in continuation of authorised leave is a condoned interruption that does not forfeit past service, but suppose that the manager decides to grant extraordinary leave to cover the absence retrospectively. Whether this period of extraordinary leave now counts towards qualifying service depends on the procedures that are followed and what is entered in the service book.

If the extraordinary leave is granted 'on medical certificate' then it counts towards qualifying service by Rule 21 and it is enough to note this in the service book. If the grounds for the leave are 'inability to rejoin duty on account of civil commotion' then a competent authority *may* allow the period to qualify (Rule 21). According to the GIDs, at one time necessary orders had to be passed by 'an authority other than the leave sanctioning authority' but the procedure has now been simplified. This type of extraordinary leave now qualifies automatically 'without any further sanctions'. If neither of these grounds apply (and the leave is not granted for 'prosecuting higher technical and scientific studies' during the absence) the period does not count towards pension.

According to GID(1) under Rule 21 extraordinary leave on all other grounds is treated as non-qualifying. But a definite entry must be made in the service book to this effect. "Even where this is not done, it should still be possible to rectify the omission during the period for preparatory action [specified] ... At the end of that period, however, no further enquiry into past events or check of past records should be undertaken. Specific entries in the service records regarding non-qualifying periods will be taken note of and such periods excluded from the service. All spells of extraordinary leave not covered by such specific entries will be deemed to be qualifying service".

For the purpose of calculating pension entitlements at the end of the employee's career it is sufficient to check for the presence of a definite entry excluding the period of extraordinary leave from qualifying service. (Our representation also checks whether there were medical certificate or civil commotion or scientific study grounds in case the entry was made in error.) But a program that oversees the maintenance of service books would have to do more. It should inform its user of the procedure to follow ("To grant this kind of leave retrospectively, obtain clearance from authority *X* by completing form *F*"); it should not allow certain entries to made at all, or indicate that they are being made in breach of the Rules ("You cannot grant leave on medical grounds and then exclude it from pension" or "You cannot include this entry in the service book now"); ideally it would also point out the consequences of omitting or including an entry if this could affect the eventual pension entitlement ("If you do not exclude this period of leave explicitly then it will qualify for pension automatically").

Such a program would be an application of some considerable novelty and it would require attention to some interesting technical questions. The main technical requirements would appear to be:

- Some kind of representation of the deontic modalities is required, though it is impossible to say without closer examination whether the program itself would need to engage in reasoning with these modalities and what kind of deontic logic would be required. Questions have been raised about the adequacy of various deontic logics, and the role of deontic reasoning generally in legal analysis programs, but no clear answers have emerged. See for example the discussion in [Bench-Capon 1989], [Jones 1990], [McCarty 1989], [Sergot 1990] and work referred to there.
- The representation of administrative (quasi-)legal procedures has received comparatively little attention. The only attempts of which we are aware are the approaches described in [Nitta et al. 1988] and [Roberts 1988]. Of course it may be that the program does not require an explicit representation of procedures (with which it can reason) but simply the ability to present the description of a procedure in the form of a text. There are numerous examples of systems that perform the latter.
- Finally, almost all legal analysis programs (of which we are aware) have concentrated on the problem of determining whether some legal concept holds on the facts of a given case in hand. There is some technical interest in how to implement a system that would take a more active role, in this case bringing to its user's attention the (quasi-)legal consequences of adding another entry to the service book when it has been omitted.

We wish to make one further concluding remark. It has been observed before that rules and regulations and legislative provisions vary widely in their character, and that questions about the adequacy of a representation cannot be divorced from questions about what the representation is for and how it will be used. In this example like in many others, the same Rule can be read, and used, and represented in many different ways. On a superficial reading, many of the Pension Rules seem to give conditions that contribute to the definition of qualifying service. A closer reading, in this case helped by the GIDs, reveals that the conditions are not part of the definition of qualifying service but constrain rather the way that the calculations are performed and the administrative details that support them.

7. CONCLUSION AND FURTHER DEVELOPMENTS

We have described a program that computes an employee's pension entitlement according to the CCS Pension Rules in the standard reference text. This program can be regarded either as a legal analysis program (because of the techniques employed in its construction) or as a data processing application (because of the task that it performs).

There are many other possible representations of the Rules defining qualifying service, especially as regards the treatment of time. We have presented only what seemed to us to be the simplest and most perspicuous representation, where all data about an employee's Government career and the Rules defining qualifying service are formulated in terms of time periods only. An alternative approach would focus instead on the transitions between these time periods, that is on events like appointment, promotion, transfer to another post, the start of a period of extraordinary leave, resumption of duty, and so on. We have experimented with a formulation of the Rules in the style of the event calculus [Kowalski & Sergot 1986] where states or the time periods of interest are computed from the events that initiate or terminate them. The Rules can be formulated in this alternative framework, yielding a representation that arguably copes more gracefully with some of the detail. What is not clear until the exercise is complete is whether the resulting representation would be easier to explain to a Government employee or pension administrator.

An intermediate approach would leave the representation of the Rules unchanged, but would describe an employee's service history as a sequence of events in the form of a narrative. The event calculus can then be applied directly to provide an interface between the two styles of representation. This third possibility is perhaps the most promising and the most elegant, since a narrative account would correspond more closely to what actually appears in a service book. Unfortunately it is also the approach which requires the greatest re-implementation of the data entry program, which is the most tedious and time-consuming part of any implementation.

We have not discussed these various options for the treatment of time in any detail because there is nothing specifically 'legal' about them. Of more significance to the construction of legal applications specifically is the fact that our program deals with only one aspect of the Rules defining qualifying service. It calculates pension entitlement from the information in the service book at the time of retirement.

Like many regulations, the CCS Pension Rules devote as much or more attention to administrative matters. We sketched in section 6 why the representation would have to be different if we attempted to provide a program to deal with the administrative aspects of the Pension Rules; the maintenance of an electronic version of the service book is the obvious application. It is difficult to believe that a prospective client, in this case the Government pension departments, would be satisfied with a system that ignores the mass of administrative detail entirely. It is this missing third component of the system which appears to present the greatest challenge and the one which is most deserving of future attention.

ACKNOWLEDGEMENTS

We wish to express our thanks to the United Nations Development Programme (UNDP) for its support through the Knowledge Based Computer Systems project. A.S. Kamble held a UNDP Fellowship for part of this work.

REFERENCES

Bajaj, K.K., Dubash, R.K., Kowalski, R.A. [1989a] Central Government Pension Rules as a Logic Program. In Knowledge Based Computer Systems (Ramani, S., Chandrasekar, R., Anjaneyulu, K.S.R., Eds), Narosa Publishing, New Delhi, 1989, pp 19-28.

Bajaj, K.K., Dubash, R.K., Kamble, A.S., Kowalski, R.A., Murthy, B.K. [1989b] Indian Import Policy and Procedures as a Logic Program. *Proc. Third International Conference on Logica, Informatica, Diritto* (Martino, A.A., Ed). Istituto per la Documentazione Giuridica, Florence 1989.

Bench-Capon, T.J.M. [1989] Deep models, Normative reasoning and Legal expert systems. *Proc. Second International Conference on Artificial Intelligence and Law*, Vancouver, 1987 (ACM Press), pp 37-45.

Bench-Capon, T.J.M., Robinson, G.O., Routen, T.W., Sergot, M.J. [1987] Logic Programming for Large Scale Applications in Law: A formalisation of Supplementary Benefit legislation. *Proc. First International Conference on Artificial Intelligence and Law*, Boston, May 1987 (ACM Press), pp 190-198.

Hammond, P., Sergot, M.J. [1983] A PROLOG shell for logic based expert systems. In *Expert Systems 83: Proc. Third Technical Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge, December 1983 (British Computer Society), pp 95-104.

Jones, A.J.I. [1990] Deontic Logic and Legal Knowledge Representation. *Ratio Juris 3*, 2 (July 1990), pp 237-244.

Kowalski, R.A., Sergot, M.J. [1986] A Logic based Calculus of Events. *New Generation Computing 4*, 1 (Feb. 1986), pp 67-95. Also in *Knowledge Base Management Systems* (Thanos, Schmidt, Eds). Springer-Verlag, Heidelberg, 1989.

McCarty, L.T. [1989] A Language for Legal Discourse I. Basic features. *Proc. Second International Conference on Artificial Intelligence and Law*, Vancouver, 1987 (ACM Press), pp 180-189.

Nitta, K., Nagao, J., Mizutori, T. [1988] KRIP2: A Knowledge Representation and Inference System for Procedural Law. *New Generation Computing 5*, 4 (1988), pp 319-359.

Roberts, H. [1988] *Knowledge Representation for Procedural Law.* MSc thesis. Department of Computing, Imperial College, London, 1988.

Sergot, M.J. [1983] A Query-the-User Facility for Logic Programming. In *Integrated Interactive Computer Systems* (Degano, P., Sandewall, E., Eds). North-Holland, Amsterdam, 1983. Also in *New Horizons in Educational Computing* (Yazdani, M., Ed). Ellis Horwood, Chichester, 1984, pp 145-163.

Sergot, M.J. [1988] Representing Legislation as Logic Programs. In *Machine Intelligence 11* (Hayes, J.E., Michie, D., Richards, J., Eds). Oxford University Press, 1988, pp 209-260.

Sergot, M.J. [1990] The Representation of Law in Computer Programs: A Survey and Comparison. In *Knowledge Based Systems and Legal Applications* (Bench-Capon, T.J.M., Ed.). Academic Press 1990.

Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T. [1986] The British Nationality Act as a Logic Program. *Communications of the ACM 29*, 5 (May 1986), pp 370-386.

Stamper, R. [1980] LEGOL: Modelling Legal Rules by Computer. In *Computer Science and Law* (Niblett, B., Ed). Cambridge University Press, New York, 1980, pp 45-71.

MacProlog is a product of Logic Programming Associates Ltd.