INTELLIGENT SYSTEMS FOR THE FORMULATION OF BUILDING REGULATIONS

David Stone and David A. Wilcox

Building Directorate
Scottish Office
Edinburgh EH1 3SZ U.K.

1.0 Introduction

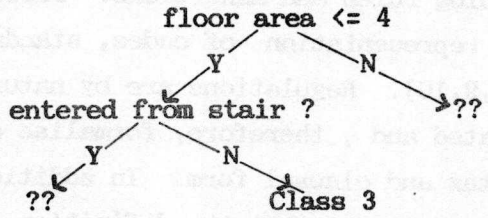
The Building Directorate of the Scottish Office is responsible for the Building Standards (Scotland) Regulations. The Regulations are intended to safeguard public health and safety and to ensure energy conservation in the built environment and set out statutory, technical requirements for building design and construction. As a part of a major review programme the Directorate has been developing intelligent, knowledge-based, systems for use by the writers of regulations for the formulation and analysis of information.

The problems inherent in drafting and maintaining large bodies of regulatory or statutory information have been well documented [1-7]. The sheer volume of information involved can itself be daunting. The current Scottish Regulations occupy just over five hundred pages of text and the current revision process has involved contributions from seven authors. The administrative problems of co-ordinating contributions, amending and revising requirements following consultation and arranging, indexing and cross-referencing material can be formidable. Until recently [3,4], there have been few examples of attempts to use formal methods to assist in this process of specifying and testing proposed regulations. The absence of an acceptable, systematic framework for the development of material can lead, despite the best intentions of authors, to regulations which are, at best, idiosyncratic in style and, at worst, to regulations with in-built inconsistencies. Fenves et al[5,6,7] identified a range of such inconsistencies which can appear in regulation texts. These include conflicting

requirements, incomplete rule sets, redundant conditions and circularities in definition. One of the most common of such inconsistencies is an incomplete requirement. Consider, for example, the following fragment from the fire regulations :-

> "..the walls and ceilings of rooms having a floor area not more than 4 sq.m. and not entered directly from a stairway enclosure may have a surface classification of 3 ...."

This at first reading appears to be an unambiguous requirement. It is, however, not explicit on what is required when the floor area is greater than 4 sq.m. or the room is entered directly from a stairway. A representation of the requirement as a decision tree reveals that two additional rules are required :-



It can be shown that these kinds of inconsistencies are syntactic or logical properties of information and can be considered independently from its semantics or meaning. The significance of this is that, given a suitable representation, the process of checking information for syntactic inconsistencies can be automated.

These problems of drafting regulations texts can have consequences for their use in practice. The large volume of material to be searched, especially where it is poorly arranged, often frustrates easy access to information. Idiosyncracies of drafting style can obscure an authors meaning and intent whilst inadvertent errors and inconsistencies can lead to differing interpretations of requirements by users and enforcing authorities.

In the work described in this paper we have been attempting to address some

of these problems of drafting and maintaining bodies of regulatory texts by developing a suitable, logic-based, computing environment for standards writers. We had a number of objectives in developing such an environment. Firstly, we wanted to devise a systematic framework for developing regulations information which ensured an orderly accumulation of material and a consistent style of presentation. Secondly, we wanted the system to act as an intelligent critic of texts in terms of an ability to detect syntactic errors. Finally, we wanted the output from this writer system to be in a format that could be used directly in end-user applications such as expert systems.

The system described has been implemented in a version of the logic programming language Prolog and centres upon a formalisation of regulation texts in a database of Prolog rules and assertions. Prolog offers a number of advantages for the representation of codes, standards and statutory information in general [8,9,10]. Regulations are by nature definitional in character and rule orientated and , therefore, formalise quite readily into Prolog's English-like syntax and clausal form. In addition, the inherent modularity of Prolog clauses means that the definition of information can proceed incrementally; general, top-level, requirements can be defined in advance of the rules governing their lower-level, detailed, conditions. In the context of regulations this permits the progressive refinement of information during the formative stages of defining a requirement.

The value of Prolog, however, is not confined simply to a convenient representation of regulations information. Prolog's in-built computational strategy of logical inference enables the logical consequences of a particular formulation of a regulation requirement to be examined in a convenient and systematic manner. By addressing a query to the system and, where necessary, supplying a range of values for design variables, the behaviour and effect of a set of requirements can be observed. This simulation of the use in practice of a proposed regulation can indicate whether the intent of a set of requirements has been properly conveyed in the expression of the rules.

There is a final advantage that can be argued for a Prolog representation of regulations. The work cited earlier [Fenves ibid] defined a set of

properties which regulatory information should exhibit if it is to be considered properly expressed and free of inconsistencies. Fenves proposed a range of formal representations for information which could be used as a basis for testing for these required properties. These included decision logic tables, decision trees and information networks. Prolog's list structured clausal form naturally implements the logical features of these formal representations. Prolog clauses can, therefore, themselves be analysed to ensure the syntactic properties of a formalisation.

It is, however, unrealistic to expect users to communicate with a system directly in terms of Prolog syntax and clauses. We have, therefore, developed an interface which enables a writer to communicate with the system using conventional English texts. The system provides a set of structured forms for entering texts which are based upon an analysis of the content and organisation of existing regulations information; this structure of regulations information and the forms are described in section 2. The texts entered on the forms are parsed by the system and mapped into an equivalent set of Prolog rules and assertions; this process and the parser are described in section 3. The system provides, in addition to the normal data management functions for editing, searching and sorting material, a set of functions for the analysis of the syntactic properties of the information and for the display of its logical structure; these are described in section 4. Finally, section 5 describes a system for end-users which makes use of the output from the writer environment and which provides the normal expert system functions of intelligent information retrieval, consultative dialogue and explanation of reasoning. A summary of the functions of the environments described is shown schematically in Figure 1.

2.0 Structure of regulations information

The primary objective of building regulations is to ensure the health and safety of people in and around buildings. The development of a statutory requirement to meet this objective proceeds, ideally, through three levels of information. These levels are, statements of intent , knowledge of principle and the statutory requirements themselves. Statements of intent are essentially policy statements and are declarations of the particular

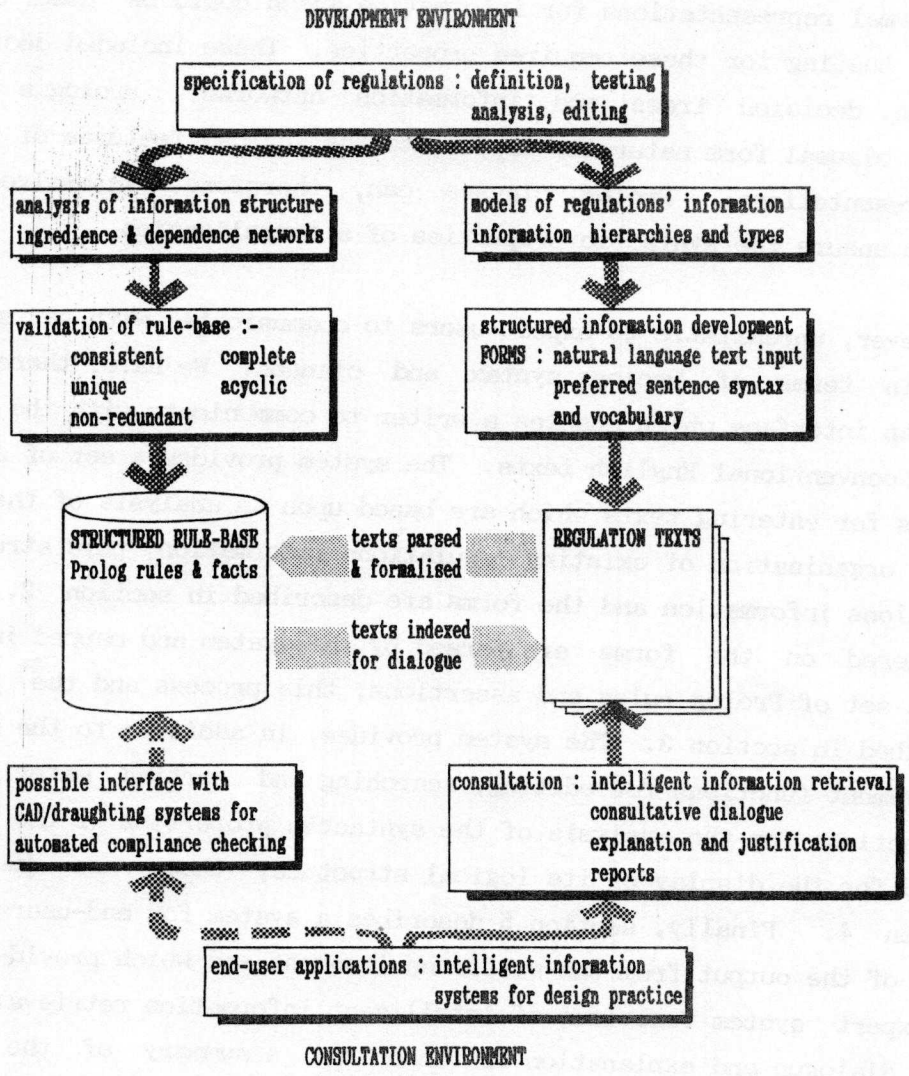


Figure 1 : Summary of system functions

risk to be avoided or the condition to be assured. Knowledge of principle involves defining the corpus of knowledge , whether derived from research, empirical observation or simply good practice, which provides the basis for translating the policy intent into an effective and enforceable requirement. The statutory requirements, the end product of this process, define how the intent can be satisfied in terms of specific attributes of physical parts of buildings or their spaces.

For a number of reasons detailed information on the supporting principles of regulation requirements is rarely explicit in existing documentation.

The work described here, therefore, has focused on the intent and requirement levels of regulations information. We nevertheless believe that knowledge of principle would be a necessary component of any comprehensive regulations knowledge-base and essential to support satisfactory and meaningful explanations in expert systems. The descriptions which follow, however, omit discussion on the representation of principle.

The primary divisions of regulations information described, then, are statements of intent and how these may be satisfied by a set of requirements. Typically, each requirement defines a required property for a given building element or a required relationship between elements. This organisation is essentially hierarchical in that it proceeds from a general objective to detailed requirements. Thus any one intention may address a range of elements and generic properties. In turn, each property may address a range of specific requirements. For example, the intent of providing adequate means of escape from fire is satisfied by taking into account, amongst other things, the width of escape routes and the width of escape stairs. The requirements for the width of escape routes are met by determining a minimum width for any one escape route and an aggregate width for all escape routes.

In addition to these primary divisions of regulatory information further sub-divisions or types of information can be identified. An important type is information which in some way qualifies the primary intent or effect of requirements. This qualifying information may be of two types. Firstly, there is information which defines when a requirement applies and, secondly, there is information which defines when an exception is permitted. Other secondary information types include definitions of the meaning of terms used in requirements and methods of measurement for properties. This limited set of information types seems adequate to characterise the bulk of regulations information. It is not restricted, however, and it is anticipated that other information types may be identified which augment or extend the set. Figure 2 shows the regulations information hierarchy described.

The following section describes how this typology of regulations information is used as a basis for the organisation of the user interface to the

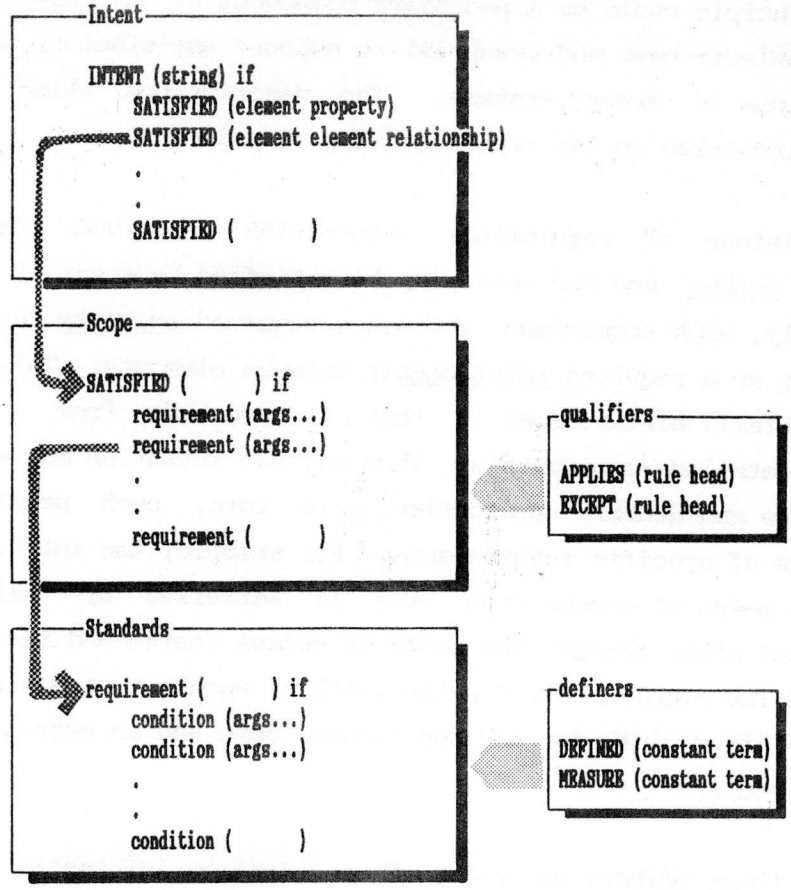


Figure 2 : Regulations information hierarchy

system and to provide a systematic framework for the development of information.

2.1 System forms

The device adopted to control communication between user and system in terms of the creation and editing of regulation texts is that of the structured form. These are intended to be analogous to conventional forms on paper and are structured to capture a particular level or type of information. For example, the most commonly used form in the system is that for entering the detailed requirements and appears on the terminal screen as in Figure 3. The head of the form is concerned with information on the intention to which the requirement refers and the building element and generic property or relationship which it addresses. The system ultimately uses

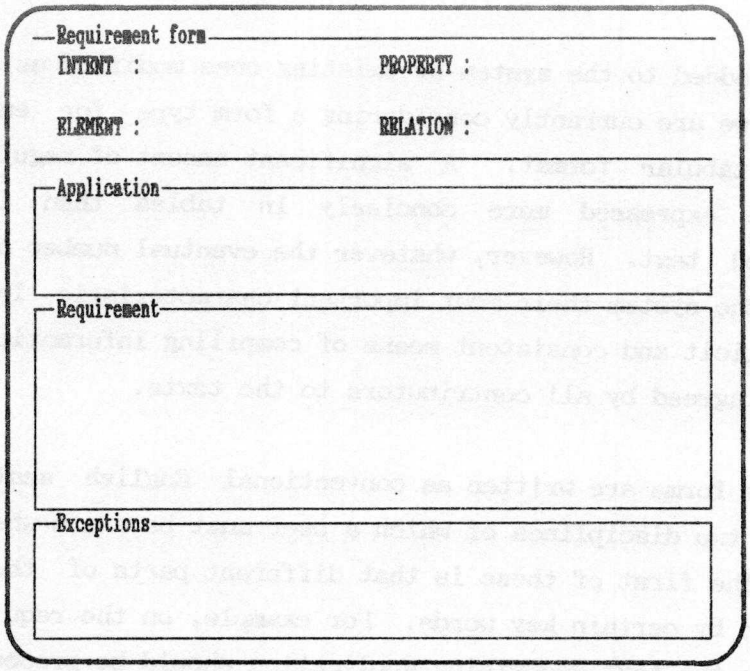


Figure 3 : Typical system form

this information to locate the requirement at the appropriate place in the internal information structure. If no information is entered here the system simply assumes that the requirements define conditions to some higher level rule. The body of the form is for the text of the requirement itself and any qualifying texts which define application or exception.

It is not necessary to enter all relevant information on a form at one session. The requirement can be refined or added to progressively. Similarly, rules of application or exception can be composed after the requirement. In this way the forms can be regarded as convenient notepads for incrementally defining information in a systematic and orderly fashion, each form contributing material to the overall information hierarchy. The user's logical view of the information he creates is always in terms of these system forms. Thus if information is recalled for editing it is recovered on the same form and in the same format in which it was entered.

At the moment the system recognises two further form types in addition to the requirement form described above. These are for entering texts on definitions of terms and methods of measurement respectively. Other forms

may be designed and added to the system or existing ones modified as necessary. For example, we are currently considering a form type for entering information in a tabular format. A significant amount of regulations information can be expressed more concisely in tables than in the equivalent, extended text. However, whatever the eventual number of form types available in the system their most important characteristic is seen as providing an explicit and consistent means of compiling information in a format which can be agreed by all contributors to the texts.

Texts entered on the forms are written as conventional English sentences. There are, however, two disciplines of which a user must be aware when composing his texts. The first of these is that different parts of the text must be delineated by certain key words. For example, on the requirement form described above the text describing application should be preceded by the words "where" or "when". Similarly the requirement text should be delineated by an "if.....and.....then" construction and any exception text preceded by the phrase "except where". Simple facts do not require any such key word delineation.

The second discipline is more subtle and is concerned with the expression of requirements. This is influenced by the system parser which reads the texts from the forms and maps them into Prolog clauses. In order to discuss this issue of expression further it is necessary to describe the strategy and behaviour of the system text parser.

3.0 System text parser

The design of the text parser exploits the fact that regulation texts are concerned with a defined domain of information that exhibits a specific information structure and typology and employs a relatively restricted vocabulary. The parser is, then, a syntactic parser. Its understanding of sentences is solely in terms of their structure or grammar. Three components in texts are recognised by the parser; word patterns , phrases , and sentences. The parser contains a set of rules of grammar which determine how the word patterns may be assembled into phrases and phrases into sentences.

The simplest component of a sentence, word patterns, are contained in a system lexicon. Nine word pattern types are recognised and these are listed below together with an identifying mnemonic and examples.

1) Determiner	(DET)	the, every, a
2) Preposition	(PRP)	of, in, from, on
3) Element	(ELM)	room, escape route, door
4) Facet	(FAC)	side, part, lining
5) Descriptor	(DES)	insulated, dangerous
6) Predicate	(PRD)	occupant capacity, travel distance
7) Operator	(OPR)	is a, is less than
8) Value	(VAL)	5, 100, yellow
9) Units	(UNT)	mm, metre

Plurals are not stored in the lexicon. The parser recognises plurals in texts and searches the lexicon only for the singular form.

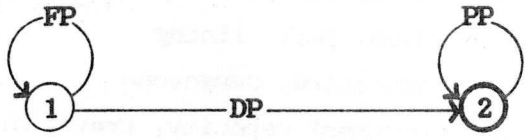
Word patterns may be assembled into three primary and three secondary phrase types. The three primary phrase types are element phrases (EP), predicate phrases (PrP) and value phrases (VP). Element phrases have three secondary phrase types. These are prepositional phrases (PP), descriptive phrases (DP) and facet phrases (FP). This set of phrase types reflect the specific character of the regulations domain which, as noted earlier, is primarily concerned with assigning property or relationship values to building elements.

The following description of the rules of grammar, whereby word patterns may be assembled into phrases and phrases into sentences, adopts the conventions suggested by Graham[11]. In this the grammar rule for an element phrase is written :-

EP --> (FP*) DP (PP*)

where the parentheses indicate that the item is optional and the asterisk * that any number of the item may occur. Thus an element phrase may consist of any number of optional facet phrases followed by a descriptive phrase followed by any number of optional prepositional phrases. The grammar

rules can, alternatively, be represented as transition networks in which nodes represent states in a parsing sequence and arcs possible transitions from one state to another. In this representation the element phrase above would appear as :-



Using this particular rule, for example, the phrase :-

"the side of the heating appliance next to the wall"

would be parsed as :-

<the side of (FP)> <the heating appliance (DP)> <next to the wall (PP)>

The complete set of grammar rules is shown in Figure 4, both in the rule form and as transition networks.

The top level of text which the parser recognises is the sentence, which consists of an assembly of phrase types. The parser currently identifies ten sentence types in regulation texts. These are listed below in the grammar rule format :-

```
S(1,2)     --> PrP EP (EP) VP
S(3,4)     --> EP (EP) PrP EP
S(5,6,7,8) --> EP (EP) PrP (VP)
S(9,10)    --> VP PrP EP (EP)
```

Text entered on a form is read and passed to a form processor which adopts the following strategy. Firstly, the text is split into its constituent sentences which are defined by the keyword delimiters. Each sentence is then passed seperately to the parser. Initially the parser generates a set of hypotheses about a sentence which involves matching word patterns in the sentence with entries in the system lexicon. More than one hypothesis may

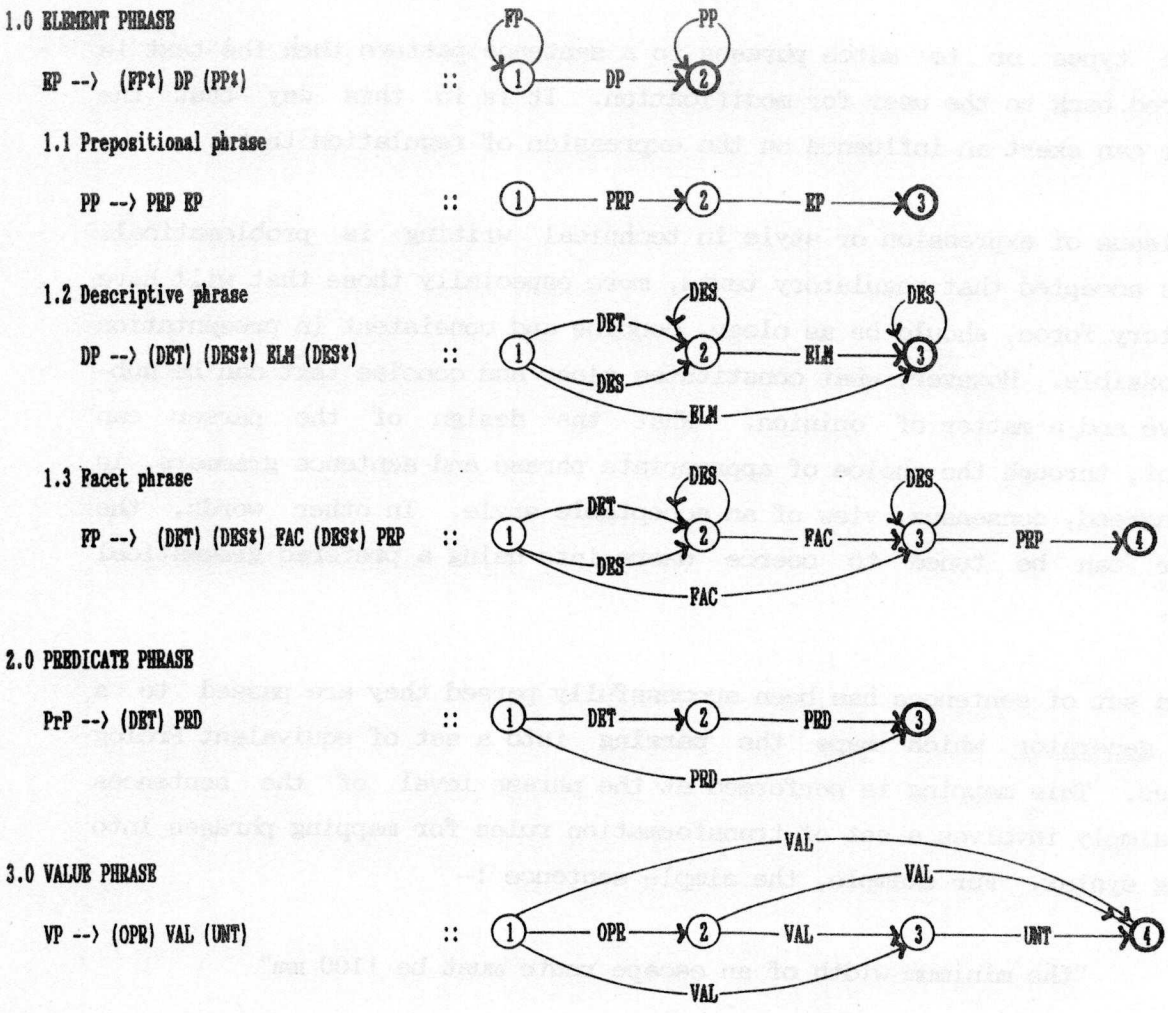


Figure 4 : Phrase grammar rules

be formed for any sentence for two reasons. Firstly, word patterns in the lexicon can be made up of one or more words, any of which may appear in more than one pattern. Secondly, words may appear in more than one category in the lexicon; "brick" for instance can be an element or a descriptor.

Once the hypotheses are formed, the parser attempts to combine the word patterns , using the phrase grammars, into a set of phrase types. In the same way, once a set of phrases is established, the parser tries to match the set to one of the sentence types in order to verify that a legitimate sentence has been found. If the parser fails either to find a set of

phrase types or to match phrases to a sentence pattern then the text is referred back to the user for modification. It is in this way that the parser can exert an influence on the expression of regulation texts.

This issue of expression or style in technical writing is problematical. It is accepted that regulatory texts, more especially those that will have statutory force, should be as clear, concise and consistent in presentation as possible. However, what constitutes clear and concise text can be subjective and a matter of opinion. What the design of the parser can reflect, through the choice of appropriate phrase and sentence grammars, is some agreed, consensus, view of an acceptable style. In other words, the parser can be tuned to coerce users into using a prefered grammatical style.

Once a set of sentences has been successfully parsed they are passed to a rule generator which maps the parsing into a set of equivalent Prolog clauses. This mapping is performed at the phrase level of the sentences and simply involves a set of transformation rules for mapping phrases into Prolog syntax. For example, the simple sentence :-

"the minimum width of an escape route must be 1100 mm"

parses to the phrase set :-

<the minimum width (PP)><an escape route (EP)><must be 1100 mm (VP)>

and is transformed into the Prolog assertion :-

(minimum-width(escape route)(1100 mm))

The Prolog clause is added into the rule-base and, where appropriate, the necessary linkages to the intent and satisfied levels in the rule-base hierarchy are instantiated. The original text from the form is added to the text files as a list of sentences, indexed from the Prolog rule-base. A schematic representation of the text input process is shown in Figure 5.

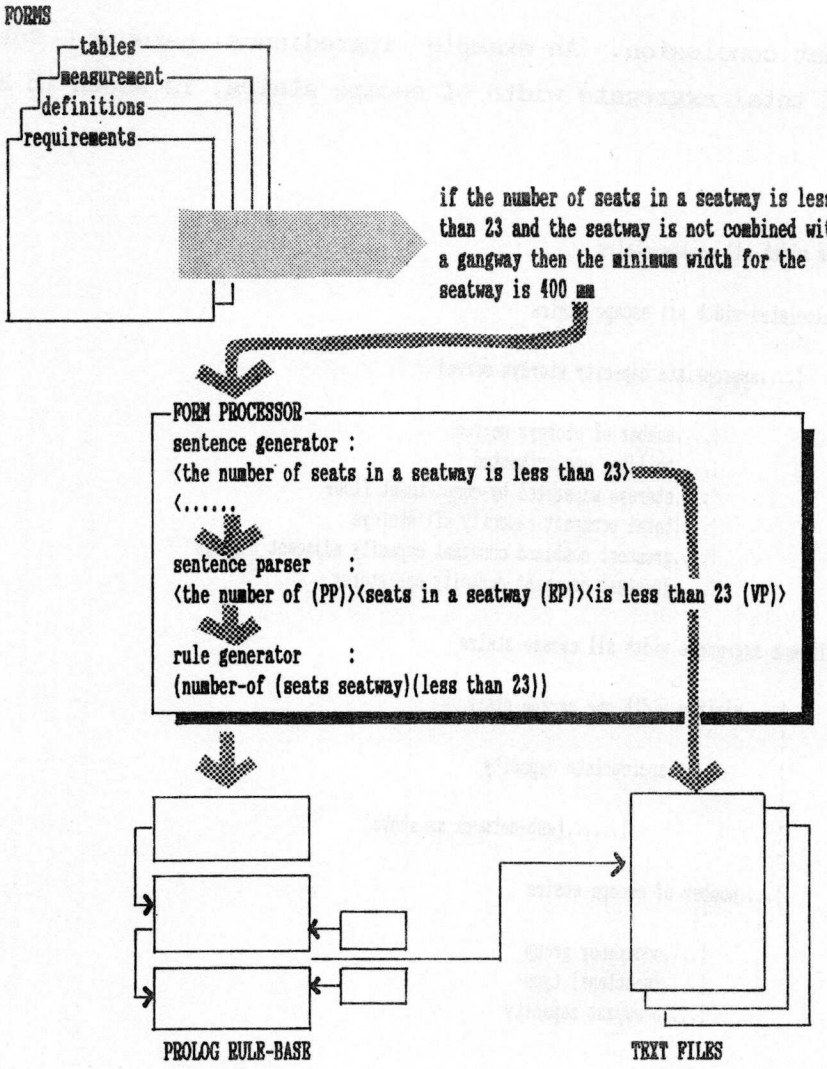


Figure 5 : Text input process

4.0 Analysis and validation of rule-sets

Once text has been entered into the system, parsed and mapped into the Prolog rule-base, the system offers a number of functions for examining the structure of the information and for ensuring that it is free from syntactic inconsistencies. These analytic and validation functions are carried out on the Prolog formalisation of the texts. The logical structure of the information can be examined in terms of two types of network, ingredience networks and dependence networks [Fenves ibid]. An ingredience network shows, for any given conclusion, all the items of information which

contribute to that conclusion. An example ingredience network, for the determination of total aggregate width of escape stairs, is shown in Figure 6 below.

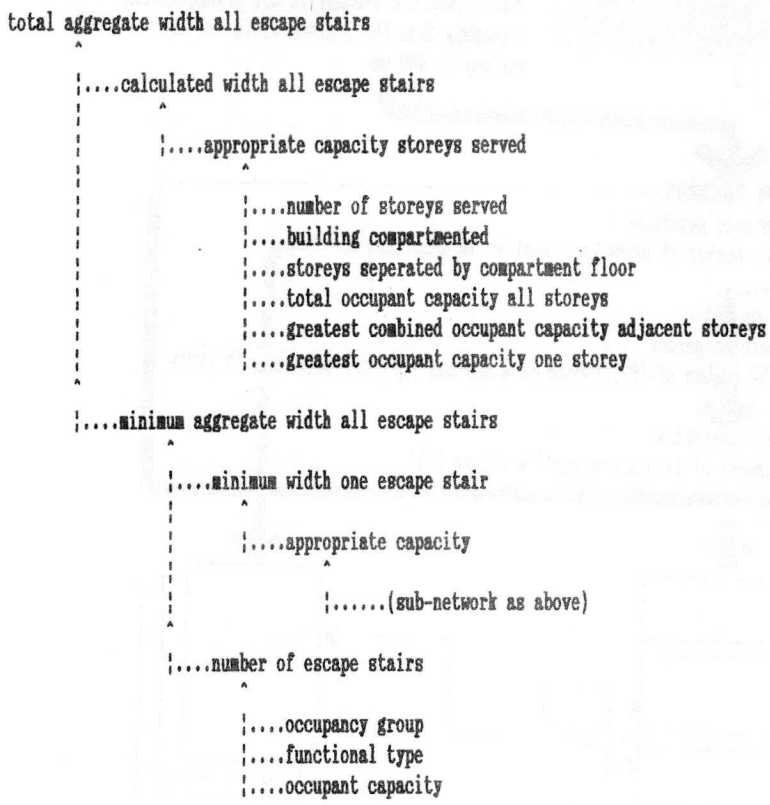


Figure 6 : Example ingredience network

Dependency networks are, in a sense, the reverse of ingredience networks. Whereas ingredience networks scan down an information tree, dependency networks show particular paths up through the tree. That is to say, dependency networks show, for any given item of information, all the possible conclusions to which it contributes. For example, information defining the occupancy classification of a building contributes to many different requirements in the regulations. Occupancy group, therefore, determines many dependency paths through the information network.

There are a number of advantages in this kind of analysis of the structure of regulations information. Ingredience and dependency networks show, graphically and in summary form, the logic and information flows implicit

in a body of text. This understanding of the structure of information can be of considerable value when arranging, indexing and cross-referencing texts in conventional documentation. By arranging texts in the order suggested by an ingredience network, for example, a reader can be progressively and logically introduced to the relevant information.

4.1 Validation

In addition to an analysis of the structure of the information the system includes several functions for determining the syntactic or logical properties of information. The particular properties of interest are that the information is :-

- consistent
- complete
- unique
- acyclic
- non-redundant

System checks for consistency are concerned with ensuring that, in the writing of requirements, a consistent terminology has been used. However, a complete determination of consistency almost always requires some semantic understanding of the text. For example, is a reference to "...internal wall surfaces" intended to be the same as a reference to "...internal surfaces of walls" ?. Given such semantic ambiguities there is a limit to the analysis a solely syntactic consistency checker can achieve. The strategy of the checker is, then, as follows. Given any requirement conclusion, the checker scans up through the information hierarchy looking for possible matches. A possible inconsistency is found when there is similarity but not an equivalence in the clause patterns. Thus the example cited above would be detected and reported as a possible error.

The properties of completeness and uniqueness in rule sets are complementary. Both are a consequence of there being variable values involved in the resolution of a requirement. In regulations these variables are design or context dependent and involve data items such as occupancy group, occupant capacity, heights of storeys, floor areas and so forth. A set of

requirements can be shown to be complete if, for any given set of variable values, the rule set always generates a conclusion. This is the problem referred to in the introduction and illustrated as an incomplete decision logic tree. The system strategy for checking for completeness is as follows. Rules are examined to detect possible branchings in their logic. The rule-base is then scanned to check that rules with the appropriate branchings exists. For simply negated conditions the check is relatively simple. Other possible logical branchings are more difficult. For example, the condition :-

(occupant-capacity (room)(greater than 100))

implies the alternative :-

(occupant-capacity (room)(less or equal 100))

The completeness checker resolves this by maintaining a dictionary of branching pairs which record value operators and their logical opposites.

Determination that a rule set is unique is a more difficult problem. A set of requirements can be said to be unique if, for any possible set of variable values, the rule set generates one, and only one, conclusion. However, the possible combinations of values for even a small and bounded set of condition variables can be large and formal testing for uniqueness, therefore, computationally prohibitive. We have adopted a pragmatic approach, therefore, to the problem of uniqueness which exploits Prolog's inherent back-tracking strategy on failure. A formalisation of a set of requirements is given a range of probable values to design variables. For any given set of values, once a conclusion has been generated, the system is forced to back-track and seek other solutions; if the formalisation is unique, no further conclusions should be generated.

Circularity and redundancy in rule sets are simpler to define and test. Checks for circularity involve scanning a rule hierarchy to ensure that it terminates and is not a closed loop in which lower level rule conditions reference higher level conclusions. Redundancy can be be characterised as a problem of over definition of requirements and is defined as follows.

Where two rules have the same conclusion and differ in respect of only one condition, then that condition is immaterial; it may, therefore, be ommitted and the rules merged.

5.0 Consultation environment : an end-user application

The development environment described above for the drafting of building regulations generates a database of regulation texts together with a corresponding formalisation of the requirements in a Prolog database of rules and assertions. These databases may then be used as the basis for the development of systems for use by end-users of regulations. Such systems may be expert system applications or, possibly, systems for automatic compliance checking interfaced to CAD/draughting packages. This section describes one such system which has been developed for the interrogation of the regulations and the determination of requirements. The system described has some of the characteristics of an expert system, that is to say, it provides consultative dialogue and explanation and justification of conclusions.

It is envisaged that users of this consultative system will include the writers of regulations themselves. It was argued earlier that one of the distinctive advantages of a Prolog formalisation of a set of regulation requirements is that, when interrogated, it reveals the logical consequences of a particular formulation. This ability to observe, as end-users, the effect of a set of requirements is seen as an important part of the testing and validation process. It is, moreover, a form of testing which cannot easily be achieved with conventional texts and documentation.

The system developed and described here is menu driven. The user indicates his interest either in terms of a subject , say means of escape from fire, or an element , say escape routes, or an element property or relationship , say the width of escape routes or the arrangement of gangways and auditoria seating. This user choice is composed internally into a query and addressed to the Prolog rule-base. All internal reasoning within the system is performed on the Prolog rules whilst all communication with the user is conducted using the original input texts. Thus it is not necessary to

compose special dialogue texts or templates.

The process of resolving the query in the database relies upon Prolog's normal backward chaining strategy. Any rule condition for which there is no match in the rule-base is assumed to be a design variable and a value is solicited from the user. At any point in a dialogue the user may ask the system for supplementary information. He may, for instance, ask what a valid answer is to a system query, or how a particular term is defined or how a particular value is measured. Once the initial query is resolved the system presents a report to the user. This contains the applicable requirement together with any exceptions that are permitted and any other relevant information that has been determined in answering the query. The user may then choose to modify any of his answers and iterate through the query process to examine alternative conclusions. Alternatively, the user may request an explanation of the requirement. System strategy for explanations is based upon the usual inference chain display , that is a progressive stepping down of a trace of the proof path [13]. It is recognised that this is an imperfect strategy for explanations, in that it provides the how but not the why of a conclusion [14]. It is our view that for building regulations the additional representation of the principles of requirements is necessary to support more satisfactory explanations. A typical dialogue with the system described is shown in Figure 7.

6.0 Summary

The paper has described a prototype intelligent system for the formulation of building regulations. The system is intended to provide a systematic environment for the development, analysis and validation of regulatory texts. Texts are entered into the system via a series of standard forms which partition regulations information into its constituent parts. Texts are parsed by the system, using preferred grammar rules, and formalised as a set of Prolog rules and assertions. The consequent rule-base can be used directly in end-user applications such as expert systems. A typical advisory system for the determination of requirements has been described.

[system is checking all requirements for escape route widths]

SYSTEM is it true that there is an escape route on a ground storey ?
USER yes
SYSTEM is it true that the escape route is part of an escape route from an escape stairway ?
USER why ?
SYSTEM I need to know if the escape route is part of an escape route from an escape stairway in order to determine if the requirement for the minimum width of the combined part of the escape route applies : is it true that the escape route is part of a route from an escape stairway ?
USER yes
SYSTEM what is the actual width of the escape stairway ?
USER valid ?
SYSTEM checking the minimum width for the escape stairway : is it true that the escape stairway serves only one storey ?
USER yes
SYSTEM what is the occupant capacity of that storey ?
USER 120
SYSTEM the minimum width for the escape stairway is 1100 mm : what is the actual width of the escape stairway ?
USER measure ?
SYSTEM the width of escape stairways is measured over the strings : handrails may intrude up to 100 mm over the stair : what is the actual width of the escape stairway ?
USER 1250 mm
SYSTEM the minimum width for the combined part of the escape route is 2300 mm

Figure 7 : Typical dialogue in consultation environment

7.0 References

1. B.N. Lewis, I.S. Horabin and C.P. Gane, Flow charts, logical trees and algorithms for rules and regulations, Centre for Administrative Studies Occasional Paper No. 2, Her Majesty's Stationery Office, London, (1967)

2. B.N. Lewis, Decision logic tables for algorithms and logical trees, Centre for Administrative Studies Occasional Paper No. 12, Her Majesty's Stationery Office, London, (1970)

3. F.I Stahl, R.N. Wright, S.J. Fenves and J.R. Harris, Expressing standards for computer-aided building design, Journal of Computer-Aided Design 15 , (6), pp. 329-334, (1983)

4. J.S. Gero et al., AMUBC system final report, Computer Applications Research Unit, Dept. of Architectural Science, University of Sydney, (1984)

5. S.J. Fenves, K. Rankin and H.K. Tejuja, The structure of building specifications , NBS Building Science Series No. 9, National Bureau of Standards, Washington D.C. (1976)

6. S.J. Fenves and R.N. Wright, The representation and use of design specifications, NBS Technical Note 940, National Bureau of Standards, Washington D.C., (1977)

7. J.R Harris, Logical analysis of Building Code provisions, Proc. 1st NBS/NCSBCS Joint Conference on Research and Innovation in the Building Regulatory Process, National Bureau of Standards, Washington D.C., pp. 285-316, (1976)

8 M. Sergot, Representing legislation as logic programs, Department of Computing, Imperial College of Science and Technology, London, (1985)

9. T. Bench-Capon and M. Sergot, Towards a rule based representation of open texture in law, Department of Computing, Imperial College of Science and Technology, London, (1985)

10. M. Sergot, Logic programming and its application in law, Department of Computing, Imperial College of Science and Technology, London, (1985)

11. N. Graham, Natural language processing, in Artificial Intelligence, pp. 209-219, Tab Books (1979)

12 G. Ritchie, H. Thomson, Natural language processing, in Artificial Intelligence ed. T. O'Shea, M. Eisenstadt, pp. 358-388, Harper & Row (1984).

13. D. Waterman et al., Expert systems for legal decision making, Expert Systems, 3 , (4), pp. 212-226, (1986)

14. D.C. Berry and D. Broadbent, Expert systems and the man-machine interface: the user interface, Expert Systems, 4 , (1), pp.18-27, (1987)