A fuzzy legal reasoner for university decision support

Dharmendra Sharma[1], Saten Behari[2]

[1]*School of Information Sciences and Engineering*
University of Canberra, Canberra ACT
Sharma@ise.canberra.edu.au
[2] *Monet Technologies Pty Limited, Sydney NSW*
behari_saten@hotmail.com

ABSTRACT

Deciding on university student disciplinary cases against stipulated rules and case facts an interesting problem for artificial intelligence. Reasoning with fuzzy rules and facts adds to the problem complexity. This paper discusses the various characteristics of the problem and presents a design and an implementation of a prototype that is modeled as a fuzzy expert system. Some test results are presented and the experience gained from the project is discussed. Some future work is also suggested to further strengthen the prototype to include a formal case specification and interaction language, and the possible drawing and use of relevant information from a knowledge base of previous cases.

Keywords: *Fuzzy expert system, artificial intelligence*

1 INTRODUCTION

An expert system is a software that models human experts in some specific domain [0]. A typical expert system comprises of general knowledge base (facts and rules), case-specific data (clusters of information), inference engine, explanation subsystem, knowledge base editor and user interface. In real life, experts are mostly faced with challenges of making a decision 'on the fly' and usually, not all of the data needed to make a sound decision may be available, some may be suspect, and some of the knowledge for interpreting the data may be unreliable or themselves incomplete. The problem of drawing inferences from uncertain or incomplete data has confronted researchers and expert system developers giving rise to a variety of technical approaches. Some useful and typical mechanisms to deal with inexact reasoning involve approximate implication, possibility theory, certainty theory and fuzzy logic as discussed in Durkin (1994), Hayes-Roth *et al* (1993) and Negotia (1985). All these methods depend on the formalization of additional *meta* knowledge in order to correct the data, take back assumptions or combine evidence. The availability of this meta knowledge is a critical factor in the viability of these approaches to particular applications.

A number of legal expert systems attempt to implement various models of reasoning by simulating a lawyer's approach to a legal problem (Popple, 1996). In contrast, the practicality of a legal expert system very depends very much on statistical approaches in reasoning which allows for quantification of vague terms thereby attaching mathematical semantics to the decision making process.

The main motivation for the work reported in this paper is need for a decision support tool that helps in an objective, fair, sound and consistent decision-making on cases that fall under student discipline regulations in a typical university situation (University of the South Pacific, USP (1998) has been used for the study). The cases have been dealt by a Discipline Committee but due to the complexity of cases, inexact nature of the regulations and the provided facts, and the changing composition of the committee, experience has shown that the rulings have differed for similar cases. The domain is small but quite intricate and provides a good technical challenge for reasoning with the fuzzy knowledge. This paper reports on a fuzzy expert system shell (called *UniLR*) designed to help model the process of decision making for case rulings. UniLR is developed as a prototype to demonstrate the underlying concepts of the problem. It has an open architecture and designed in a general way so that it can be used in like domains for legal reasoning. For further information on fuzzy logic the reader is referred to the references listed at end of article.

2 THE PROBLEM

The USP Discipline Regulation governs the staff and student behavior on campus and stipulates discipline measures for persons breaching the code of conduct. A Discipline Committee receives submissions from members of the community on any breach, identifies the appropriate charges and lays them on the accused, seeks written statements from the accused, interviews the accused for defence, checks the gathered information against the regulations, and announces a verdict of guilty, not guilty, insufficient evidence along with any penalty.

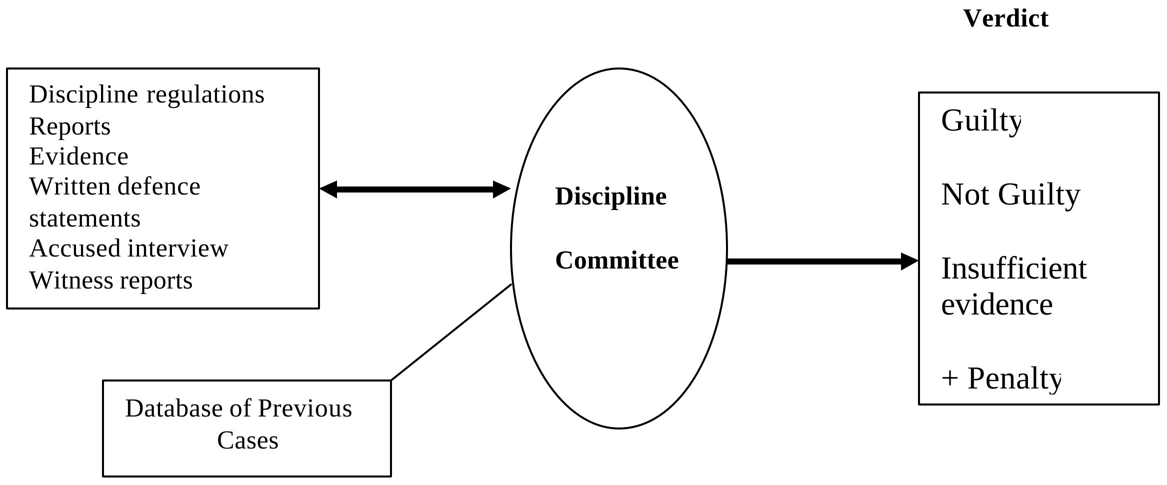

Figure 1: Discipline Committee Process

As seen from Figure 1, the Discipline Committee is responsible for laying a charge from facts from the reports statements, interview and any witness information, decide on a verdict and an appropriate penalty. The committee has to interpret the rules and the case information which is usually vague. The facts about the case and the rules are expressed in a natural language abounds with vague and imprecise concepts. The Discipline Committee's responsibility is to ensure the vague and imprecise concepts are correctly and consistently interpreted. UniLR attempts to simulate the Discipline Committee's functionality using fuzzy reasoning and guides the process to a sound decision-making. It also records facts (including fuzzy knowledge) on cases and decision structures for future use.

3 FUZZY LOGIC

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth values between "completely true" and "completely false".

Fuzzy logic is determined as a set of mathematical principles for knowledge representation based on degrees of membership rather than on crisp membership of classical binary logic Zadeh (1984).

It is primarily concerned with quantifying and reasoning about vague or fuzzy terms that appear in our natural language. In fuzzy logic, these fuzzy terms are referred to as *linguistic* or *fuzzy* variables. A fuzzy set assigns membership values between 0 and 1 that reflect more naturally a member association with the set. Let X be the range of possible values of fuzzy variable, with elements of X denoted as x. A fuzzy set A of X is characterized by a membership function $m_A(x)$ that associates each element x with a degree of membership value in A: $m_A(x): X \rightarrow [0,1]$. Event or element x is assigned a membership value by a membership function m. The value represents the degree to which element x belongs to fuzzy set A : $m_A(x) = Degree(x \in A)$ where, $0 \le m_A(x) \le 1$. For a discrete set of elements, a fuzzy set can be represented through the use of a vector: $A = (a_1, a_2, \cdots, a_n)$ where, $a_i = m_A(x_i)$. The fuzzy set theory has defined operations on its linguistic variables and rules like the classical set theory. Fuzzy logic has been used directly in many applications. Expert systems have been the most obvious recipients of the benefits of fuzzy logic, since their domain is often inherently fuzzy. Examples of expert systems with fuzzy logic central to their control are decision-support systems, financial planners, diagnostic systems for determining soybean pathology, and a meteorological expert system in China for determining areas in which to establish rubber tree orchards. Another area of application, akin to expert systems, is that of information retrieval. To date, there has been little work reported on legal applications using fuzzy logic.

4 THE DISCIPLINE REGULATIONS

In general, there are two kinds of law: *Case law* (decisional) in which the decision is made in court, *Statute* (definitional) which is determined by the legislation as stated in Sergot et al (1986). UniLR is designed to cater for both these kinds of law. For UniLR prototype development, we use the USP Discipline Ordinance and Regulations for Students (1998). It contains both kinds of rules and regulations: definitional and decisional. The knowledge acquisition, interpretation and formalization has been done with some help from legal experts. The regulations pertaining to discipline considered for UniLR are described below.

- Campus regulations – regulations to enforce discipline on campus. For example, campus regulation C14 states: *Students, whenever they are on campus, shall have in their possession at all times, a valid student ID card and shall produce the same to an officer of the University on Demand.*
- Residential regulations – regulation to see that discipline is maintained in Halls of Residence. For example, residential regulation R3 states: *Pets, animals or birds are not allowed in the Halls of Residence.*
- Assessment and examination regulations – regulations to see that honest practice prevail in
- assessment and examinations. For example, assessment and examination regulation 1.9 states: *No Candidate is to bring with him into the examination room any written or printed matter except: (a) as authorised by the examiner; or (b) where such written or printed material has been authorised for use in an approved open book examination.*

A legal case is declaratively expressed as a story of *n* sentences. The main objective is to prove the truth or falsity of each sentence in the story, which in turn determines the guilty status of filed charges. Each of the sentences in the case story has its own search space to its truth or falsity with some attributed degree of confidence. A case can have many charges, a charge can be proven to be guilty [G], not guilty [NG] or have not sufficient evidence [NSE]. A sentence has *supports* as evidence and a set of *askables* for further interrogation the user to elicit more information. The problem-solver is tasked with traversing a search space for a sentence to prove its truth or falsity.

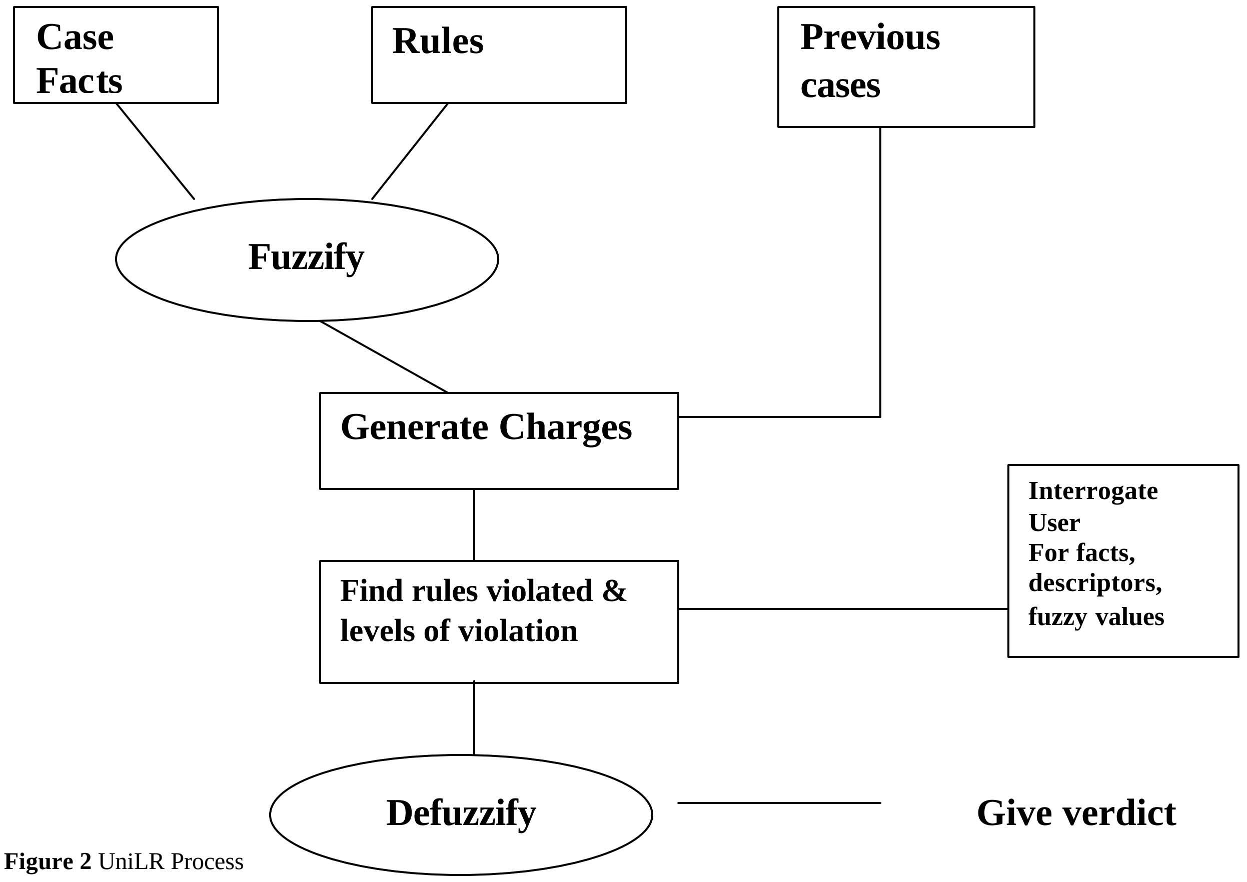
Case Facts
Rules
Previous cases
Fuzzify
Generate Charges
Interrogate User For facts, descriptors, fuzzy values
Find rules violated & levels of violation
Defuzzify
Give verdict

Figure 2 UniLR Process

5 UniLR

5.1 Architecture

UniLR makes use of a knowledge base consisting of regulations and judgmental rules of the underlying legal system, derived from the knowledge engineering process applied on the domain. Ideally, any attempt to rid the vagueness of our natural language is chimerical. Hence, UniLR uses fuzzy logic *as a formal systematic approach to inexact reasoning to give judgement at the end of the case investigation and interrogation process. Its core functionality is to give ruling on charges triggered by a given case story and justify its decision during the judgment delivery process. Figure 1 is amended to give the following functional components for UniLR.* The UniLR shell consists of charge generator, interrogator and fuzzy reasoner and includes other components as described below.

5.2 Functional Components

1. Charge Generator:
   - Picks a case sentence.
   - If sentence is valid then produce charge by the violated regulation i.e. cluster the fact by violated regulation else write to error file. (This is because of the notion of closed world assumption – the system will not recognize any sentence that is not defined.) Repeats the above process for each of the specified case facts.
2. Interrogator:
   - Picks a charge with its matching case facts.
   - Interacts with the user.
   - Uses offence justification rules and Fuzzy Reasoner to determine whether guilty.
3. Fuzzy Reasoner:
   - Provides reasoning service to the Interrogator based on fuzzy logic system.
   - Stores the result of the current case in a file to support future learning while using case based matching.
   - Displays the results at the end of the interrogation process.

5.3 Knowledge Base

UniLR's knowledge base consists of:

1. Case story (facts) – information on a given case.
2. Rules(coded) – expresses the rules and
3. regulations of the underlying legal system and used to come up with charges during the charge generation process.
4. Judgmental rules – initiates expression of knowledge of a expert with the quest for evidence/support for the sentence/offence in consideration. Based on the result of search for support/evidence, these rules may trigger *askables* or other *jrules*.
5. Askables – are questions used to extract any additional information from the user.
6. Regulations (text) – specifies the text version of the regulations from the application domain.
7. Penalty – expresses the penalties assigned to each of the rules.
8. Previous case results - contains verdicts from previous cases.

5.4 Knowledge Representation

The UniLR shell currently uses knowledge represented as Prolog clauses. The <*keyword*> used in the knowledge base can be functors to attach additional semantics to a case fact or evidence. Specific format for each component of the knowledge base is summarized as follows.

1. A case story consists of the following prolog clauses

   student(<id>) – *specifies the student being accused*

   st(<stid>,<stkeyword>,'<sentence>') – *expresses each sentence in the case story that has violated a rule. The <stkeyword> should be found in one of the rules. <sentence> is the text version of the <stkeyword>.*

   su(<stid>,<sukeyword>) – *expresses support/evidence for each encoded sentence. <stid> specifies the link between a support and a sentence. <sukeyword> should be found in one of the jrules. The use of <id> allows same support text to be used for different sentences.*
2. Rules

   rule(<rule id>,<list of keywords>) – *where the <list of keywords> should be used to express the case sentences.*
3. Judgemental rules

   jrule(<jrule id>, G, NG, NSE, ID, <sukeyword>, X, F) – *where the <jrule id> uniquely identifies each of the judgmental rules. G, NG, NSE are the fuzzy variable which gets membership values assigned on the result of the quest for the desired evidence/support. ID identifies the sentence being processed, X caches the result of the evidence search, and F specifies link of this jrule to other jrules and/or askables.*

4. Askables
   ask(<ask id>,G , NG, NSE, A, F) - *where the <ask id> uniquely identifies each of the askables. G, NG, NSE are fuzzy variables which gets membership values assigned on the response of the user. A caches the user response, and F specifies link of this askable to other jrules or askables*
5. Regulations – text version of rules
   regulation (<rule id>, '<regulation text>') – *where the <rule id> identifies its corresponding rule*
6. Penalty
7. **penalty(<rule id>, <offence number>, '<penalty text>')** – *where the <rule id> identifies its corresponding rule* Case result – at the end of the case processing, the information of the entire case is captured as a single Prolog clause in the following format. This caters for future matching with previous cases, extending the UniLR to a hybrid of rule-based and case-based expert system shell.
   Case(<student id>, [[<rule id>, <AG>, <ANG>, <ANSE>, <list of sentences violating>, <rule id>>, <list of supports and responses to askable>]
   [<rule id> ,…………]…….])

where the <rule id> identifies a charge and <AG>, <ANG>, <ANSE> specify average membership values per charge to each of the fuzzy sets on which the ruling is based.

An element in the <list of sentences violating rule id> is the following format:

[<sentence id>,<stkeyword>,'<sentence>']

An element in the <list of supports and responses> is in the following format:

[<guilty membership value> <not guilty membership value> <not sufficient membership values> <response/support id><response/sukeyword>]

~ in front of the sukeyword means support not found

5.5 Meta-knowledge

Some of the meta knowledge/rules used in UniLR shell are as follows:

1. Rule in which the defendant takes plea gets priority over other rules at any point in time. For example, at time *t* the conflict set contains rules R11, R12, R13 and out of which R12 is the rule which determines the plea of the defendant then R12 will take priority during conflict resolution.
2. Definitional rules do not require interrogation. For example, Examination Regulation 1.9
3. states that: *No Candidate is to bring with him into the examination room any written or printed matter except: (a) as authorised by the*
4. *examiner; or (b) where such written or printed material has been authorised for use in an approved open book examination.*If a student is caught using unathorised material in the exam then he is automatically guilty and the system can be used to find out the penalty guidelines.
5. Since each of the sentences has its own support and askables (i.e. a separate search space), the initial conflict set can be resolved in any order as all the applicable rules are valid at the initial problem solving point. The current version of the *UniLR* shell selects the first rule in the conflict set based on the default order of rules in the prolog database, in other words, the rules are loaded in an ordered fashion with respect to its importance to the problem, the shortest possible solution path. However, further conflict resolution should select first rule in the conflict set (list) as the fired rule changes the conflict set.
6. At the end of the interrogation, if decision is not clearer (low maximum membership value [<60%] from the fuzzy reasoner) then the Rule to get the opinion of the user is fired. Depending on the response, this alters the decision state of the system before the rule was fired else the system adheres to its previous decision state.

5.6 Search

The search is primarily depth first with backtracking done during case investigation process. It uses forward chaining through symbolic pattern matching. The shell is implemented in such a way that it contains two different phases of search and both of them are depth-first with backtracking. In the first level of search, the UniLR shell traverses through the encoded case story of *n* sentences $<S_1, S_2, S_3,\ldots\ldots, S_n>$ and the encoded regulation $<R_1, R_2, R_3,\ldots\ldots, R_p>$ generating *m* charges $<C_1, C_2, C_3,\ldots\ldots, C_m>$ for the case story where $m £ n$. This search is very simple as it traverses through the search space and determines which regulations has been violated in the case story. In the second level of search, UniLR shell traverses the search space for each charge interrogating in a depth first manner. At each node of the search tree, the shell either quests for an evidence or fires an askable to get a judgmental decision from the user. The search tree for each of the charges under microscope is dynamic and is created during the interrogation process from the responses and evidence encountered. Based on the result of each node, the shell assigns a membership value to each of the fuzzy sets *<G, NG, NSE>*. The shell traverses *m* sub trees to determine guilty status of each of the *m* charges.

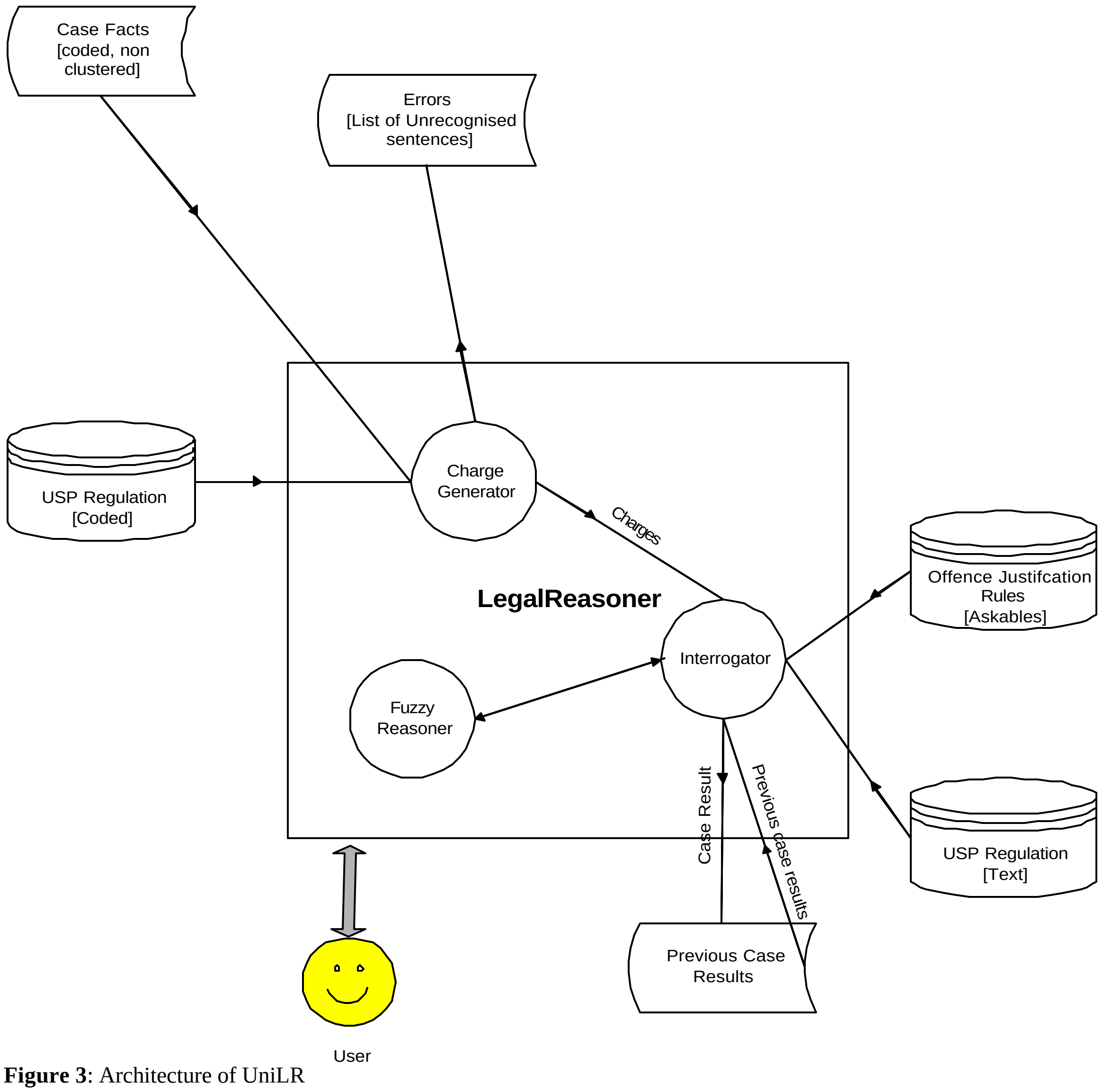

Figure 3: Architecture of UniLR

5.7 Membership Value Assignment

The shell is very flexible in membership value assignment for the fuzzy variables. It allows membership values to be assigned during the knowledge engineering process. This caters for context dependency of the legal system being used. It also overcomes the problem of deep inferencing while using certainty factor as discussed in Durkin (1994). Further to this, it provides for integration of multiple experts' knowledge, through manipulation of membership values by the rules of fuzzy algebra. Rules of fuzzy algebra involves the notion of concentration, dilation, intensification, power, intersection, union, addition, subtraction, multiplication, division, and complementation of membership values and linguistic variables. Due to nature of problem and generic approach in building the shell, the semantics of problem solving has been taken into consideration. The membership values assigned to the fuzzy variables exhibiting the degree of belonging, is a variable for each interrogation depending on the context of the sentence involved. For example, a <*nq*> *not quite* response for question 1, an expert may mean weight of 0.3 to Fuzzy Set *A*. On the other hand, a <*nq*> *not quite* response for question 2, the expert may mean weight of 0.1 to Fuzzy Set *A*. This dilemma is conquered by the use of the mentioned knowledge representation format. In contrast, with the utilization of the mentioned membership approach, the question of quantification of responses and evidence comes into picture. As the

UniLR Shell needs the membership values to be coded during the knowledge engineering stage, a rule of thumb (Heuristic) can be used by the knowledge engineer to quantify its askables and evidence. For example, if *Guilty* fuzzy variable is assigned 0.8 for a response, then based on the nature of the question and response, the *Not Guilty* fuzzy variable could be assigned a value that is the complement of the *Guilty* fuzzy variable. However, this varies from evidence to evidence and question to question. Hence a general heuristics for membership values assignment will have problems in reasoning the quantification of domain specific knowledge there by rising the complexity of knowledge engineering process. This calls for the heuristic to be response and evidence dependent, although similar responses can use same heuristic in membership values assignment to the fuzzy variable.

5.8 Vagueness and Fuzzy Reasoning

UniLR implements fuzzy reasoning in judgement delivery process. It uses a linguistic variable called *CHARGE. CHARGE* has three fuzzy sets, *GUILTY [G], NOT GUILTY [NG], NOT SUFFICIENT EVIDENCE [NSE]*, defined on it which captures partial membership of a response and/or evidence to the fuzzy variable. Hence, → $m_A(x) = Degree(x \in A)$, where $0 \le m_A(x) \le 1$, *A* is *G, NG* and *NSE, X = charge.* At the end of the case investigation process the fuzzy reasoner does the average membership value, AMV_A, for each of the fuzzy set to come with a decision of AMV_A degrees of confidence. The highest AMV_A, of each of the fuzzy sets determines the guilty status of the charge under the microscope:

$$MVA_A = \frac{\sum_{i=1}^{n} A_i}{n} \quad where\ A = G, NG, NSE\ and\ n = no.of\ evidence/askable\ for\ each\ A \cdot$$

6 UniLR ALGORITHM

1. System Initialization
2. Charge Generator

```
BEGIN Charge Generation
  COLLECT all the sentence id’s in list l
  WHILE list l is not empty
       GET the head of list l
       GET the corresponding sentence
       VALIDATE the sentence
       IF Valid THEN
            FIND the regulation violated by the sentence
            CLUSTER sentence by violated rule
            ;one rule can be violated by many sentences
       ELSE
            WRITE the sentence id to the error file
       END IF
  END WHILE
END
```

3. Interrogator

```
BEGIN Interrogation
  WHILE not end of charge list CL
       SELECT a charge C from CL
       WHILE not end of sentence list SL for CL
            SELECT a sentence S from SL
                ;SL is list of sentences violating the charged ;rule. For n
                charges there will be n SL of ;varying size.
                COLLECT all the applicable jrules giving AR
                ;AR is the conflict set on SL
                WHILE conflict set RA is not empty
                 SELECT the first rule R in AR
                 ;conflict resolution
                 FIRE rule R
                      COLLECT response/evidence with assigned membership
                      values of R
                      IF R triggers other rules then
                            PREPEND the new applicable rules to the
                            conflict set RA
                 END IF
                 RA = RA – R
                END WHILE
            SL = SL – S
        END WHILE
        CL = CL – C
    END WHILE
END
```

4. Fuzzy Reasoner

```
BEGIN fuzzy reasoning
    WHILE not end of charge list CL
        SELECT charge C from CL
          COLLECT all the membership value for guilty fuzzy variable G
          COMPUTE average of membership $AMV_G$ for G
        STORE $AMV_G$ with C
          COLLECT all the membership value for not guilty fuzzy variable NG
          COMPUTE average of membership $AMV_{NG}$ for NG
        STORE $AMV_{NG}$ with C
          COLLECT all the membership value for not sufficient fuzzy
          variable NSE
          COMPUTE average of membership $AMV_{NSE}$ for NSE
        STORE $AMV_{NSE}$ with C
        CL = CL - C
    END WHILE

    WHILE not end of charge list CL
        SELECT charge C from CL
        PERFORM fuzzy reasoning based on $MVA_C$
        DISPLAY judgement
        DISPLAY justification of the judgement – I/O
        CL = CL - C
    END WHILE
END
```

7 CONCLUSION

UniLR is a successful realization of fuzzy logic in legal decision-making. The problem had been abstracted, conceptualized and solved using the fuzzy logic metaphor. A prototype has been developed to exhibit the practicality of the researched concept at both abstract (conceptual) and implementation (concrete) level with respect to the problem nature/domain (legal case) and problem solving paradigm (fuzzy logic). Fuzzy reasoning has been successfully implemented in the shell. Although, the utilized rules of fuzzy algebra could change with respect to a particular application within the problem domain, the methodology of realizing fuzzy logic metaphor will remain the same. Further, the implemented knowledge representation format is rich enough to cater for easier extension of the shell from rule based to a hybrid of case based and rule based legal expert system shell. Also, the shell is generic in nature separating the problem-solving engine from the problem and domain specific knowledge and hence, can be used with any legal system that supports decisional and/or definitional nature of cases.

Further work on the shell includes development of problem specific heuristics for membership value assignment to the fuzzy variable; support for *why* question during the interrogation process; matching with previous case results either using pure symbolic pattern matching and/or values of fuzzy variables to incorporate learning into shell; and English like natural language user interface. An induction subsystem needs to be developed to learn from previous cases (including the linguistic patterns) and to use the patterns to facilitate the processing of new cases.

REFERENCES

Baldwin, J. F. 1981. Fuzzy logic and fuzzy reasoning. In: E.H. Mamdani and B.R. Gaines (eds.). *Fuzzy Reasoning and Its Applications*. London: Academic Press.

Durkin J. 1994. *Expert Systems – Design and Development*. Macmillan Publishing Company. New York.

Hayes-Roth, F., Waterman, D. A. and Lenat, D. B. 1993. *Building Expert Systems. Vol. 1.* Addison-Wesley Publishing Company, Inc. Canada.

Le, T. V. 1993. *Techniques of Prolog Programming.* John Wiley & Sons, Inc., Canada.

Li, H. and Gupta, M. 1995. Fuzzy logic and Intelligent Systems. Kluwer Academic Publishers, Boston.

Mellish C. S. and Clocksin W. F. 1994. *Programming in Prolog. 4th Ed.* Springer-Verlag. New York.

Merritt, D. 1989. *Building Expert Systems in Prolog.* Springer-Verlag, New York.

Negoita, C. V. 1985. *Expert Systems and Fuzzy Systems.* The Benjamin/Cummings Publishing Company Inc., California.

Popple, J. 1996. *A Pragmatic Legal Expert System.* Aldershort Dartmouth, U.K.

Puppe, F. 1993. *Systematic Introduction to Expert Systems – Knowledge representations and problem solving methods.* Springer-Verlag, New York.

Radecki, T. 1982. An evaluation of the fuzzy set theory approach to information retrieval. In: R. Trappl, N. V., Findler, and W. Horn (eds.). *Progress in Cybernetics and System Research, Vol. 11*. Proceedings of a Symposium organized by the Austrian Society for Cybernetic Studies. Hemisphere Publ. Co., N.Y.

Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwackzek, F., Hammond, P., and Cory, H. T. 1986. The British Nationality Act As A Logic Program. *Communications of the ACM*. **29** (5).

Shoham, Y. 1994. *Artificial Intelligence Techniques in Prolog.* Morgan Kaufman Publishers Inc., U.S.A.

University of the South Pacific Discipline Regulations, 1998.

Zadeh, L. A. 1984. Fuzzy sets. *Information and Control* **8**, 338-353.

Zadeh, L. A., 1984. Making computers think like people. *IEEE Spectrum* **8**, 26-32.