International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-2, Issue-7)

Research Article | **July 2013**

Implementation of RETE Algorithm Using Lemon Expert System

M. Veera Narayana[1], A. Sunil Kumar[2]
Assistant Professor
Dept of CSE & IT
KIMS College of Engg.
Amalapuram-533201, India

B. Suneel Kumar[3] ,
Assistant Professor
Dept of CSE & IT
CIET
Rajahmundry-4, India

N.Samkishan[4] , B. Jogeswara Rao[5]
Dept of CS & SE
AUCE (A)
Andhra University
Visakhapatnam-03 , India

Abstract—

The RETE algorithm is an efficient well-organized pattern matching algorithm for implementing production rule systems. The present paper focuses on the RETE algorithm to support disjunctive relationships using "or" connectivity and use the shadow proxy mechanism to update objects in the working memory. A Lemon Expert System is developed to identify the diseases of lemon crop to diagnose the diseases which are affected to the Lemon plants. This system works on the mechanism of Rule based system and the RETE Algorithm. Programmed interviews with domain experts are conducted to build a Lemon Expert Knowledge base. A separate user interface consisting of three different modules namely, End-user/farmer, Expert and Admin are presented here. End-user/farmer module may be used for identifying the diseases for the symptoms entered by the farmer. Expert module used for adding rules and questions to data set by a domain expert. Admin module is used for maintenance like database recovery, when failure occurs, adding new hardware components to improve the quality of the system etc. This expert system is a web based application for online users with JSP as front end and MYSQL as backend.

Keywords- Expert Systems, Pattern Matching algorithm, RETE algorithm, Lemon, JSP and MYSQL.

I. INTRODUCTION

A. Pattern Matching

Pattern matching [17 and 18] is the act of checking some sequence of rules for the presence of the constituents of some pattern. The match usually expected to be exact. The patterns generally are of the form of either sequences or tree structures. Pattern matching include outputting the locations (if any) of a pattern within a token sequence, to output some component of the matched pattern, and to substitute the matching pattern with some other token sequence (i.e., search and replace). Sequence patterns (e.g., a text string) are often described using regular expressions and matched using techniques such as backtracking. Tree patterns are used in some programming languages as a general tool to process data based on its structure, e.g., Haskell [21], ML and the symbolic mathematics language Mathematica [17] have special syntax for expressing tree patterns and a language construct for conditional execution and value retrieval based on it. For simplicity and efficiency reasons, these tree patterns lack some features that are available in regular expressions.

B. Expert Systems

An expert system [12] is a computer system that emulates the decision-making ability of a human expert, i.e. it acts in all respects as a human expert. Expert systems have emerged from early work in problem solving, mainly because of the importance of domain-specific knowledge. The expert knowledge must be obtained from specialists or other sources of expertise, such as texts, journal articles, and data bases. Expert system receives facts from the user and provides expertise in return. The user interacts with the system through a user interface, constructed by using menus, natural language or any other style of interaction. The rules collected from the domain experts are encoded in the form of Knowledge base. The inference engine may infer conclusions from the knowledge base and the facts supplied by the user. Expert systems may or may not have learning components. A series of Expert advisory systems [12], [13], [14] were developed in the field of agriculture and implemented in www.indiakisan.net[14].

C. RETE Algorithm

The RETE algorithm [1, 2 and 3] was designed by Dr. Charles L. Forgy in 1974 in Carnegie Mellon University and first published as a working paper. The rule engines may provide specialized support within the RETE network in order to apply pattern-matching rule processing to specific data types and sources such as programmatic objects, XML data or relational data tables. RETE algorithm is much simpler to directly use the common objects as facts in rule engines. The rule engine mentioned in this paper handles common objects as facts to provide easy integration. RETE networks [6] are directed acyclic graphs that represent higher-level rule sets. Rule sets are normally represented at run-time using a network of in-memory objects. These networks match rule patterns to actualities (relational data tuple). RETE networks act as special type of relational query processor, performing projections, selections and joins conditionally on arbitrary numbers of data tuple. They are collected into rule sets which are then translated into an executable RETE network [7] in rule compilation phase. Each working memory element (WME) enters the RETE network at a single root node. The root node passes each WME on to its child nodes, and each WME propagates through

the network, possibly being stored in intermediate memories, until it arrives at a terminal node. The network has two parts one was Alpha and another was Beta networks. Alpha network performs constant tests on working memory elements and the outputs are stored in alpha memory (AM). The beta network contains the joins of networks of networks and beta memories. Join nodes perform the tests for consistency of variable bindings between conditions. Beta memories store partial instantiations of productions.

II. KNOWLEDGE BASE

Expert system knowledge base contains a formal representation of the information provided by the domain experts. The information is collected from the domain experts by conducting programmed interviews. The Knowledge Base of the Lemon Expert System contains encoded form of rules. Each rule describes the diseases and the symptoms for corresponding diseases and the cure measures for diseases. The different symptoms and diseases occurred in lemon crop are represented in tabular format as below.

Table 1: Knowledge Base for Lemon crop

S. No	Symptoms	Disease	Cure
1.	Damage is superficial and does not affect internal fruit quality	Melanose	The fungus is controlled with fungicides. Applications should begin soon after runner development.
2.	The dark brown to black spots are produced which are raised and rough to touch.	Lemon Scab.	Use systemic fungicide such as metalaxyl MX L 35 or Apron XL 35 ES 3 WS, Apron 35 WP
3.	The fruit size and appearance on the age of the fruit at the time of infection	Brown Rot.	Spray mannose 2.5g copper Ox chloride 3g/liter
4.	The streaks of yellow gum and one form of stem end rot of fruit (stem-end rot)	Collar Rot.	Spray Endosulphan 5% and Phosphorus 10% to effected plant
5.	Attacks producing slightly raised, irregular scabby or wart-like outgrowths.	Botrytis (Grey Mould)	Spray Zineb/Meneb @ 2.5-4.0 g/liter of water.
6	The woody debris as stumps and dead tree roots, especially the blood wood trees.	Armillaria	Resistance in triploid seedless lemon has been identified
7	The canopy at the top and failure to form vigorous new growth. The foliage turns yellow and twigs die back.	Phytophthora Root Rot	persists many years in soil and so, lemon should not be replanted into infested soils for at least five years.

8	lemon trees become infected with honeydew-excreting insects Its may affect tree performance and fruit by interfering with photosynthesis.	Sooty Mould	Soil fumigation is an effective control, but is not usually economical. Soils with consistent problems should not be planted to cucurbits
9	The fungus survives on infected twigs, dead wood, leaves and in leaf litter. The spores are carried onto the fruit by rain splash. Light tan to brown sunken spots the fruit rind.	Septoria Spot	Excellent control of this disease can be achieved with mefanoxam (Ridomil Gold),
10	The black colouring is due to the growth of fine and lives on the surface of the fruit.	Sooty Blotch	Cracks, scarring and pitting can be caused by mechanical damage when fruits are young, plants.

This information is used to construct different rules stored in the knowledge base. Generally the rules of the form,

Rule1:
If the symptoms S1=0, S2=1, S3=0, S4=1, S5=1, S6=0, S7=1, S8=0, S9= 1, S10=0 then
Resultant disease may be "**Botrytis**".

Rule2:
If the symptoms S1=1, S2=0, S3=1, S4=0, S5=0, S6=1, S7=0, S8=1, S9=0, S10=1 then
Resultant disease may be "**.Sooty Blotch**".

Rule3:
If the symptoms S1=0, S2=1, S3=1, S4=0, S5=1, S6=1, S7=0, S8=1, S9=1, S10=1 then
Resultant disease may be "**Melanose**"

Rule4:
If the symptoms S1=1, S2=1, S3=0, S4=1, S5=1, S6=0, S7=1, S8=0, S9= 0, S10=0 then
Resultant disease may be "**Lemon Scab.**".

Rule5:
If the symptoms S1=0, S2=1, S3=0, S4=1, S5=1, S6=0, S7=0, S8=1, S9= 1, S10=0 then
Resultant disease may be ".**Brown Rot.**".

Rule6:
If the symptoms S1=0, S2=1, S3=0, S4=0, S5=1, S6=0, S7=1, S8=0, S9= 1, S10=0 then
Resultant disease may be "**Botrytis (Grey Mould)**".

Rule7:
If the symptoms S1=0, S2=1, S3=1, S4=0, S5=1, S6=0, S7=1, S8=0, S9= 1, S10=0 then
Resultant disease may be "**Collar Rot.**".
Rule8:
If the symptoms S1=0, S2=1, S3=0, S4=0, S5=0, S6=0, S7=1, S8=0,S9= 1, S10=0 then
Resultant disease may be "**Armillaria**"

Rule9:
If the symptoms S1=0, S2=1, S3=0, S4=1, S5=1, S6=0, S7=1, S8=0,S9= 1, S10=0 then
Resultant disease may be "**Alternaria**".

Rule10:
If the symptoms S1=0, S2=1, S3=0, S4=1, S5=1, S6=0, S7=1, S8=0, S9= 1, S10=0 then
Resultant disease may be "**Lemon Scab**"

Rule11:
If the symptoms S1=1, S2=1, S3=0, S4=1, S5=1, S6=0, S7=1, S8=0, S9= 1, S10=0 resultant disease may be "**Septoria Spot**".

Rule12:
If the symptoms S1=1, S2=1, S3=1, S4=0, S5=0, S6=1, S7=0, S8=1, S9=0, S10=1 then
Resultant disease may be "**Lemon Scab**".

Rule13:
If the symptoms S1=0, S2=1, S3=1, S4=0, S5=1, S6=1, S7=0, S8=1,S9=0, S10=1 then
Resultant disease may be "**Leaf Spot**"

Rule14:
If the symptoms S1=1, S2=1, S3=0, S4=0, S5=1, S6=0, S7=1, S8=0, S9= 0, S10=0 then
Resultant disease may be "**Leaf Spot**".

Rule15:
If the symptoms S1=0, S2=1, S3=0, S4=1, S5=0, S6=0, S7=0, S8=1, S9= 1, S10=0 then
Resultant disease may be "**Leaf Spot**".

Rule16:
If the symptoms S1=0, S2=1, S3=0, S4=0, S5=1, S6=7, S7=0, S8=1, S9= 1, S10=0 then
Resultant disease may be "**Lemon Scab**".

Rule17:
If the symptoms S1=0, S2=1, S3=1, S4=0, S5=1, S6=0, S7=0, S8=0,S9= 0, S10=0 then
Resultant disease may be "**Melanose**".

Rule18:
If the symptoms S1=0, S2=0, S3=0, S4=0, S5=0, S6=0, S7=1, S8=0,S9= 1, S10=0 then
Resultant disease may be "**Brown Rot**"

Rule19:
If the symptoms S1=0, S2=1, S3=0, S4=0, S5=1, S6=0, S7=1, S8=0,S9= 1, S10=0 then
Resultant disease may be "**Melanose**".

Rule20:
If the symptoms S1=0, S2=1, S3=0, S4=0, S5=1, S6=0, S7=1, S8=0,S9= 1, S10=0 then
Resultant disease may be "**Lemon Scab**"

Rule21:
If the symptoms S1=0, S2=0, S3=0, S4=0, S5=0, S6=0, S7=1, S8=0,S9= 1, S10=0 then
Resultant disease may be "**Brown Rot Disease**"

Rule22:
If the symptoms S1=0, S2=1, S3=0, S4=0, S5=1, S6=0, S7=1, S8=1,S9= 1, S10=0 then
Resultant disease may be " **Dodder Causal Agent**".

Rule23:
If the symptoms S1=0, S2=1, S3=0, S4=0, S5=1, S6=0, S7=1, S8=0,S9= 0, S10=1 then
Resultant disease may be "**Melanose**"

Implementation of RETE Algorithm:

A. RETE Algorithm

RETE algorithm is implemented based on the two criteria's: first one is Rule decomposition and the second one was shadow proxy.

Examples of Rules:

Rule"OR"
When (service(type=="A")
Or (service(type=="B")
And$person: person(memberType=="CommanMember"),
Choose ServiceType==$type)
Then
$person.setMemberType("GoldMember");
Update($person);
End.

Rule decomposition for Lemon database:

There are three conditions in the rule decomposition is
R1: Service (type == "A"),
R2: Service (types == "B"),
R3: Person (member Type =="Common Member", choose Service Type == type), and the rule could be simply described as (R1 *or* R2) *and* R3.

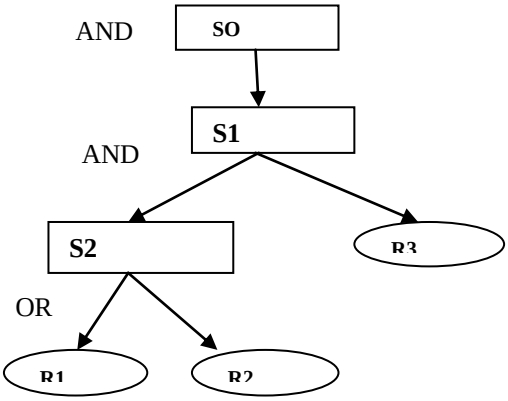

Figure 2a: Original Tree

The conditions are explicitly grouped with parenthesis or connected by logic connective (*and*, *or*) as group condition of lemon crop Symptoms abbreviated as Symptoms and the type of Symptoms is either AND or OR. Thus (R1 *or* R2) is a Symptoms whose type is OR and includes two conditions R1, R2. Additionally the root of the LHS (left hand side) is an implicit Symptoms whose type is AND, and it doesn't need to be specified. Finally the rule should be represented as Fig.2.(a) S0 represents the root Symptoms, S2 represents (R1 *or* R2) and S1 represents S2 *and* R3. Improved algorithm to transform tree 2a in to tree 2c, then decomposes the rule into two sub-rules. Fig 2. (a) S0 and S1 have the same type, so the S1 is deleted and the children of S1 are added to the S0's children, the intermediate tree is depicted in the Fig. 2.(b).

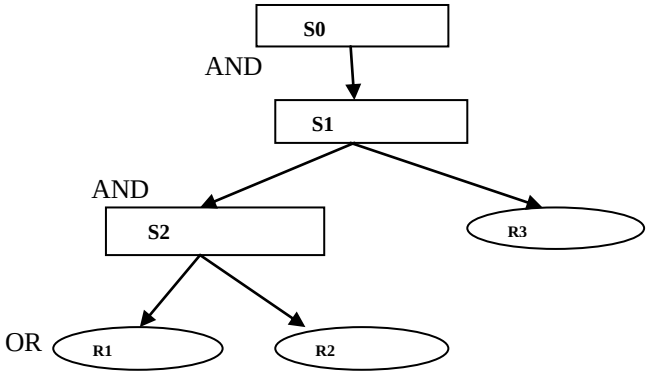

Figure 2b: Intermediate Tree

Secondly if the child node is Symptoms OF S2 and its type is OR, we need apply transformation to the parent node. Supposed that a parent node has n Symptoms children whose Type are OR represent as the SUM of or and m other children. Each node in Sum of or has M1, M2 …Mn children respectively, then there are N (= M1*M2*…*Mn) permutations. Finally we get transformed tree represented in the Fig. 2. (c), the resultant rule (R1 *and* R3) *or* (R2 *and* R3), the type of the root Symptom is OR, its children are sub-rules, so we can get 2 sub-rules R1 *and* R3, R2 *and* R3. In the rule compilation part, the RETE networks have two terminal nodes which represent the 2 sub-rules respectively.

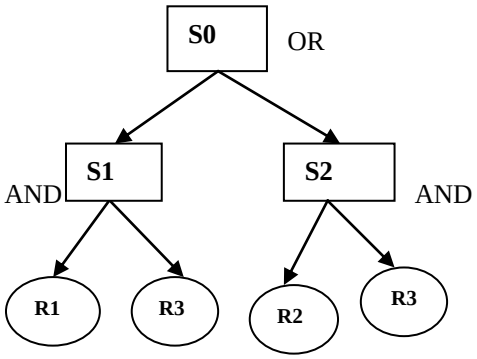

Figure 2c: Transformation Tree

The rule decomposition is done with the transform algorithm. The pseudo code for the algorithm is given below

```
TRANSFORM  (Group Condition parent)
{
STEP I: we divide children into two parts: OR and not OR, the array ors records the OR part, others record not OR.
      Group Condition ors [];
Condition
STEP II: transform the type of the parent others [the number of parent node's children];
     int permutations = 1, index = 0;
     for (each child of the parent node)
if the child is Group Condition and the type of the child is OR, we calculate the number  of the permutations
      Permutations * = the number of the child's children;
      add the child to the array ors;
  else others[index] = child;
  index++;
End
for node into OR type, clear up the children of the parent node, initialize the array indexes with value 0, the array index
will record which child of the ors[j] node will be added to the permutation.
              int indexes[the length of the array ors];
STEP III: now we know how many permutations we will have, and create permutations.
for ( int i = 1; i <= permutations; i++ )
      we create a group condition whose type is AND Group Condition temp, then we create the actual permutations;
       int mod = 1;
       for (int j = the length of the array ors - 1;j >= 0;j--)
add the indexes[j]'th child of ors[j] node to the temp's children;
       if (( i % mod) equals to 0), indexes[j] = (indexes[j] + 1) % the children number of the ors[j];
       mod *= the children number of ors[j];
End for.
//conditions originally outside OR will appear in every permutation, so add them in their original position.
   for (int j = 0; j<the length of the array others; j++ )
        if others[j] does not equal to NULL, add others[j] to the temp's children;
    End for.
     add temp to the parent node's children;
End for.
STEP IV: remove duplications.
Parent. PACK ();
}
```

2. Shadow Proxy Agent:

A shadow proxy acts as an agent of an asserted fact, and it has two attributes 'shadow' and 'shadowed'. The 'shadow' attribute is a "copy" of all attributes of the asserted fact and The 'shadowed' attribute maintains a reference to the asserted fact.

The rule engine will reason over the 'shadow' attribute of shadow proxy instead of reasoning over asserted objects directly.

If any non traceable change occurs to the common object's attributes, the engine will be shielded by shadow proxy and remain consistent.

If the application wants the changes to become "visible" to the engine, it just needs to notify the engine by calling 'update' primitive. The 'update' primitive will involve two operations

- Firstly we use the 'shadow' attribute to retract the shadow proxy corresponded with the assert fact, the process of retraction is tree-based and rematch-based removal.
- Secondly we use the 'shadowed' attribute which maintains a reference of the changed fact to reset the 'shadow' attribute.

Finally the rule engine will reason over the shadow attribute of shadow proxy again, ending display in Fig 4 proposed RETE algorithm tree of expert system. The elapsed time for different numbers of data sets in the implementation of RETE for lemon database is given in the following table.

Table 2: Execution Time of the RETE Algorithm

No. of Training **Data sets**	Elapsed Time in (Microseconds) **RETE**
	4
10	12
20	21
30	35
40	49
50	61

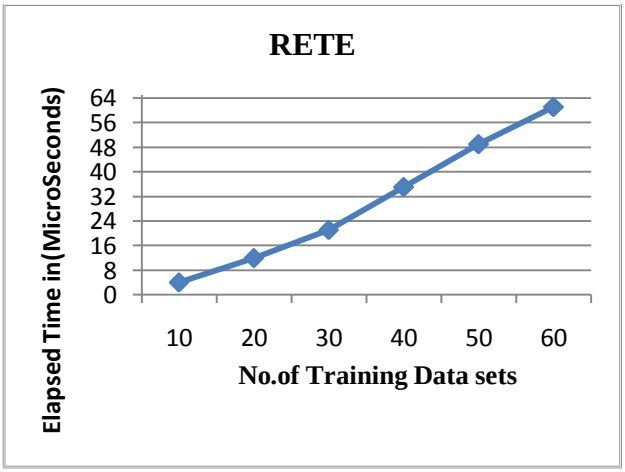

Figure3: Graph describing of RETE Algorithm

III. RESULTS:

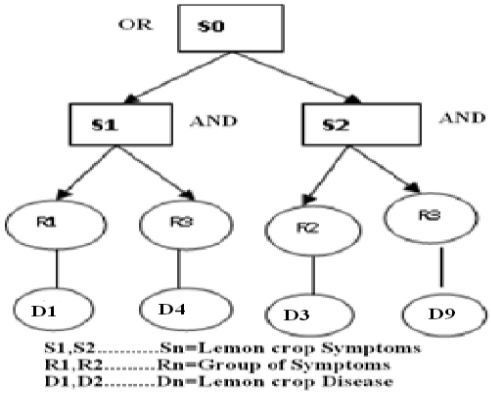

Figure 4: Proposed RETE Tree of Expert System

Description: In this screen shot, the user can submit the observed symptoms to the lemon advisory system through online by selecting the appropriate radio buttons for the processing of the symptoms observed.

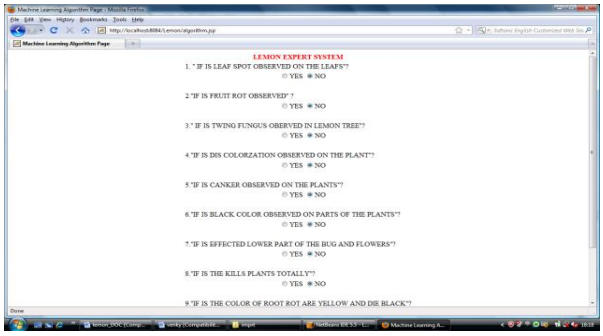

Figure 5: Screen for selecting symptoms in Lemon Expert System

Description: In this screen shot, the algorithm takes the input given by the user and generating the following advices.

Figure 6: Displaying advices to the farmer

IV. COMPARATIVE STUDY

The execution time of the improved RETE algorithm and the other pattern matching algorithms are calculated and tabulated as follows

Table 3: Execution Time of the Algorithms

No. of Training Data sets	Elapsed Time in (Microseconds) **Implementation RETE**	Elapsed Time in (Microseconds) **Naïve Matching**	Elapsed Time in (Microseconds) **ID3**
0	4	10	15
10	12	15	21
20	21	31	39
30	35	57	63
40	49	64	71
50	61	77	90

Based on the above values, a graph is drawn taking the number of training datasets on X-axis and the execution time of RETE algorithm, Naïve Matching algorithm and ID3 algorithms are taken on the Y- axis (figure 3).

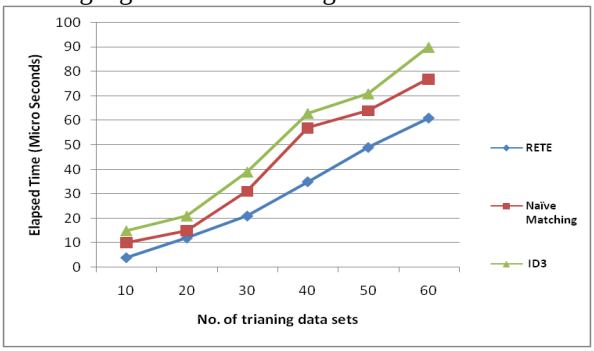

Figure7: Graph describing the performance of RETE Algorithm

It can be observed that, as the number of datasets increases the execution time of the RETE algorithm decreases much more than the other algorithms.

V. LEMON EXPERT SYSTEM ARCHITECTURE

The Proposed architecture of the Lemon Expert System consists of Rule Based Expert System, RETE algorithm and Knowledge Base which are used in the inference mechanism. It is represented in the figure 4

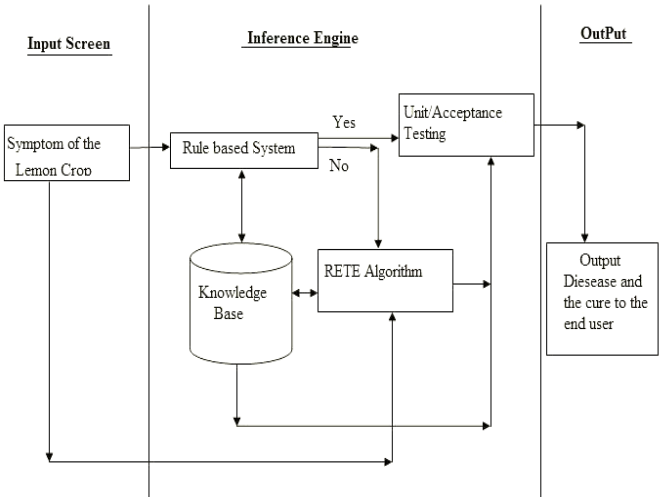

Figure 7: Proposed architecture of Expert System

The User Interface of the Lemon expert system consisting of three different interfaces namely, End-user/farmer, Expert and Admin, is presented here. End-user/farmer module may be used for identifying the diseases for the symptoms entered by the farmer. Expert module may be used for adding rules and questions to data set by a domain expert. Admin module may be used for maintenance of the system.

Figure 8: Screen for Introduction Lemon Expert System.

Figure 9: Screen for Photo gallery of Lemon Expert System

Description: In this screen shot, the user can submit the observed symptoms to the lemon rule based system through online by selecting the appropriate radio buttons for the processing of the symptoms observed.

Figure 10: Screen for Rule based of Lemon Expert System

Description: In this screen shot, the algorithm takes the input given by the user and generating the following advices.

Figure 11: Displaying Rule based to the farmer

VI. CONCLUSIONS

It is observed that as the number of datasets increases the execution time of the RETE algorithm decreases much more than the other algorithms and hence concluded that RETE algorithm shows better performance than the other algorithms. A Lemon Expert System is developed taking RETE algorithm in inference engine for providing expert advices to the lemon growing farmers. A well designed interface for giving Lemon plant related advices and suggestions in the area to farmers. This system provides facilities like dynamic interaction between expert system and the user without the need of domain expert at all times. Also this project employs common objects as facts to make rule engines easily integrate into application systems.

REFERENCES

[1]. Forg,C.L,"Rete: a fast algorithm for the many pattern/many object pattern match problem", Artificial intelligence, vol. 19. 1982, pp.17- 37.

[2]. Jean-Luc Gaudiot, and Andrew Sohn, "Data-Driven Parallel Production Systems", IEEE Transactions on Software Engineering, 3rd ed. vol.16.Mar. 1990, pp. 281-293.

[3] Robert B. Doorenbos, "Production Matching for Large Learning Systems", Computer Science Department, Carnegie Mellon University Pittsburgh, Ph.D. thesis, 1995

[4]. Apache Drools Project, "Apache Drools Expert User Guide", 2008.

[5] Wikipedia,http://en.wikipedia.org/wiki/Rete_algorithm, 2009.

[6]. Karen Walzer, Tino Breddin, and Matthias Groch, "Relative temporal constraints in the Rete algorithm for complex event detection",Proceedings of the second international conference on Distributed eventbased systems, Jul. 2008, pp 147-155.

[7] Karen Walzer, Matthias Groch, and Tino Breddin, "Time to the Rescue - Supporting Temporal Reasoning in the Rete Algorithm for Complex Event Processing", Proceedings of the 19th international conference on Database and Expert Systems Applications, Sep. 2008, pp. 635-642.

[8] Ding Xiao, Xiaoan Zhong. Improving Rete Algorithm to Enhance Performance of Rule Engine Systems for ICCDA 2010

[9] Prof. M.S. Prasad Babu, N.V. Ramana Murty, S.V.N.L.Narayana, "A Web Based Tomato Crop Expert Information System Based on Artificial Intelligence and Machine learning algorithms", *International Journal of Computer Science and Information Technologies, Vol. 1 (1), (ISSN: 0975-9646).*, 2010, pp6-15.

[10] Prof. M.S. Prasad Babu, Mrs. J. Anitha, K. Hari Krishna, "A Web Based Sweet Orange Crop Expert System using Rule Based System and Artificial Bee Colony Optimization Algorithm" , *International Journal of Engineering Science and Technology ,vol.2(6),*2010.

[11] www.indiakisan.net

[12] Prof. M.S. Prasad Babu, N.Thirupathi Rao, "Implementation of Parallel Optimized ABC Algorithm with SMA Technique for Garlic Expert Advisory System", *International Journal of Computer Science, Engineering and Technology (IJCSET), Volume 1, Issue 3,* October 2010.
[13] Prof. M.S. Prasad Babu, M. Sudara Chinna, "Implementation of Rank Based Genetic Algorithm for Cotton Advisory System", *International Journal of Computer Science and Technology, vol 9(6*), 2010.
[14] Andrew Colin, "Building Decision Trees with the ID3 Algorithm", *Dr. Dobbs Journal June* 1996.
[15] Prof. M.S. Prasad Babu, Mrs. J. Anitha, K. Hari Krishna, "A Web Based Sweet Orange Crop Expert System using Rule Based System and Artificial Bee Colony Optimization Algorithm", *International Journal of Engineering Science and Technology ,vol.2(6),*2010.
[16] Krogh, A., and Vedelsby, J., "Neural Networks Ensembles, Active Learners". *In Tesauro, G; Touretzky, D., and Leen, T., eds., NIPS-7, pp.231-238.Cambridge, MA: MIT press.* 1995
[17] The Mathematica Book, chapter Section 2.3: Patterns
[18] The Haskell 98 Report, chapter 3.17 Pattern Matching.