Asian Journal of Information Technology 4 (10): 920-926, 2005


# Training and Extracting Real Time System Symbolic Rules Using Neural Networks

Nabil M. Hewahi
Department of Computer Science,Islamic University of Gaza, Palestine

**Abstract:** Various attempts towards training nets to represent rule based systems or extracting rules from neural nets have been tried. Most of the previous study was concentrating on standard rule structure of the form IF <condition> THEN <action>. In this study two methods present to train a neural net to correctly represent censored production rules of the form IF <condition> THEN <action> UNLESS <censor>. The advantage of doing this over the standard rule structure is that censor production rules can very well serve in real time systems. Using censor production rules allow us to get more certain results given more time by checking more censors which rarely occur. One of the methods is based totally on backpropagation with a small modification. The second method is partially based on backpropagation and the rest is based on a proposed algorithm that is concerned with adjusting the net weights taking into account the importance of the censors. The weights of the links connecting the censors with the hidden layers represent the time allocated to each censor to be checked in time constraints. These weights are based on the importance of the censor and the average time for censor checking. We also present a method to extract the rules from the trained net.

**Key words:** Rule-based systems, variable precision systems, neural networks

## INTRODUCTION

One of the interesting increasing research areas in the past and current years is the combination of symbolic and connectionist approach in a one hybrid system[1-5]. Towell *et al.*[3] proposed Knowledge Based Artificial Neural Networks (KBANN) system to relax the required complete and correct knowledge bases by other standard algorithms for explanation-based learning. KBANN uses a knowledge base of domain-specific inference rules in the form of PROLOG-like clauses. The knowledge need only to support approximately correct explanations. KBANN translates the knowledge base into an Artificial Neural Network (ANN) in which units and links in the ANN correspond to parts of the knowledge base(final conclusions, facts, intermediate conclusions and dependencies)[3]. Towell and Shavlik[4] later proposed a method to extract symbolic rules from trained neural networks. The method is called MOFN algorithm and is based on two assumptions. The first assumption is that the units are either maximally active (1) or inactive(0). The second assumption is that training does not significantly alter the meaning of units. In MOFN, after the training of the network, the weights will be clustered into several equivalent classes and some unimportant classes are removed. Then the network is retrained, where the weights are fixed while only the bias of the units are changed. The experiments of the algorithms show that the extracted rules are superior to the rules produced by methods that directly refine symbolic rules and also superior to those produced by previous techniques for extracting rules from trained neural networks[4]. Fan *et al.*[1] proposed a method called Rule-based Enforced Subpopulations(RESP) for utilizing prior knowledge evolving ANNs. This methods is based on Enforced Sub Population (ESP)[2] which shown highly effective in several difficult reinforcement learning problems. RESP used KBANN to convert domain knowledge into ANN and then evolving the ANN further with ESP. RESP outperforms the ESP[1]. A new developed approach to extract symbolic rules is proposed by the Zhou *et al.*[6]. The proposed approach is called Rule Extraction From Neural network Ensemble (REFNE) and is developed to improve the comprehensibility of trained neural network ensembles that perform classification tasks. It gracefully breaks the ties made by individual neural networks in prediction. It also employs specific discretization scheme, rule form and fidelity evaluation mechanism. Experiments show that with different configurations, REFNE can extract rules with good fidelity that will explain the function of trained neural network ensembles, or rules with strong generalization ability that are even better than the trained neural network ensembles in prediction. Some other approaches that have combinations of decision trees with neural networks or utilizing neural networks in data mining and extracting symbolic rules from the trained neural networks can be found in by zhou and chen[7,8]. Most of these approaches have used standard production

**Corresponding Author:** Nabil M. Hewahi, Department of Computer Science, Islamic University of Gaza, Palestine

3rules (<IF Condition Then Action >) as the symbolic representation. Michalski and Winston[9] proposed the Censored Production Rule (CPR) to exhibit Variable Precision Logic (VPL) in which certainty varies, while specificity stays constant. The form of CPR is as follows:

**If condition then action unless censor:** Where the censor is the exception condition. Such rules are employed in situations in which the conditional statement IF condition then action holds frequently and the assertion censor holds rarely. The censors are only checked whenever time permits. The more time we have, the more censors can be checked and the more we are certain from the conclusion. If any censor holds we can not take the action of the rule. Such rules are used in real time systems. As an example to CPR.

**If working-day then john-in-office:** UNLESS John-is-sick , John-on-leave In the above example, if the time constrained is very limited, we may take the decision of John-in-office without checking any censor but we are still less certain about our conclusion. If we have more time, we may check the censors ( one or two based on time constrained) and the more censors we check, the more certain we are from the conclusion. Another thing to be noticed is that the censor of John-on-leave should have the higher importance than John-is-sick because on leave might occur more than the sickness. This should give John-on-leave higher priority to be checked before John-is-sick.

Further expansions of CPR, Bharadwaj and Jain[10] have introduced a concept of Hierarchical Censored Production Rule (HCPR). HCPR is a CPR augmented with specificity and generality information, which can be made to exhibit variable precision in the reasoning such that both certainty of belief in conclusion and specificity may be controlled by the reasoning process. Another expansion of CPR and HCPR has been introduced by Hewahi[11]. Hewahi[11] developed the General Rule Structure (GRS) which is based on the ideas of variable precision logic rules and ripple down rules developed by Compton and Richards[12]. GRS can be application independent and comprehensible (understandable). The term general used in GRS indicates that the developed rule structure takes care of generality in terms of application and reasoning process (forward and backward chaining). GRS can be used for real time applications, control applications (from the characteristics of VPL) and standard rule-based expert system applications. Based on HCPR, GRS can give more certain and specific answers, whenever time permits. GRS can be easily used in directing the system to decide, which rule should be checked next if the currently checked rule is failed. One of the main advantages of the GRS structure is the simplicity to train it; therefore, the system can determine the most commonly used rules. This would reduce the time consumed for finding the proper rule to be fired. Such a system has numerous applications in situations, where decision must be taken in real time and with uncertain information.

In present study we are stick with CPRs due to its simplicity comparing to HCPRs and GRSs. Some of the real time issues concerned with variable precision logic are in of Hewahi[13].

## RESEARCH GOAL AND USED APPROACHES

Main goal in this study is to train a neural net to represent rule based system and to have finally a neural net from which we are able to extract a set of CPRs with another form. Our attempt is to extract standard rules containing the negation of the censors that can be checked within the given time constraints. The negation of these censors should be considered as normal conditions when the system decides that they should be treated as core conditions. To make the idea clear, let us consider the CPR explained before

**If working-day then john-in-office unless john-is-sick, john-on-leave:** If the time left to check the censors is 5 time units and to check each censor we need 3 time units, then we can only check one censor. The negation of this censor should be added as a core condition to the rule and has to be the one with the higher importance. The rule to be extracted from the net will have the following shape

**If (Working-day) and (not john-on-leave) then john-in-office:** The above rule says that if it is working day and John is not on leave, then he is in his office. We notice that the censor John-is-sick is totally removed from the extracted rule. For sure if we have more time, the negation of John-is-sick can be added to the condition part of the rule.

To approach this problem, we tried two methods, the first is based totally on backpropagation algorithm with small modifications. This modification is done by fixing the weights of the censors based on time constraints and not allowing these weights to change during correcting the error backward. The second method is to apply first the backpropagation to train the net to recognize the whole set of censor production rules considering all the censors for all the rules , then we apply a special developed algorithm to adjust the net weights based on the importance of the censors and the time left to check the censors.

After applying any of the above two methods we use another developed algorithm to extract standard rules with censors that can be checked within time limits as main conditions. In both training methods we try to make weights connecting the censors with hidden layers represent the time for checking the censor. This representation is very useful in choosing which censor to be checked within time constraints. Time representation as weights is based on some factors that will be explained in the next paragraph.

It is to be known that the inputs for the neural net is the values of the conditions and censors of the system and the output is the actions of the rules. Figure 1 shows the neural net structure.

## QUESTIONS AND ANSWERS

In this section we try to answer some questions that might be asked by the reader and to make our proposed approach plausible. We shall use question answer approach which is not usually used in research to be clear as much as possible.

**Q. Why don't you use KBANN algorithm?:** KBANN algorithm assigns positive weights to positive conditions and negative weights for negated conditions, which can also be true in our proposed algorithm. But the question with KBANN, what kind of weights should be assigned to censor conditions. For sure it can not be positive or negative in terms of KBANN usage of weights. Moreover, there is no way using KBANN to recognize which censor to include and which censor to exclude in the rule to be extracted based on time limits.

**Q. Why should we use neural networks if the average time for checking a censor and the remaining time to check all the censor is known?:** The duty of neural network here is not to explore these times but to be trained through examples. We obtain a neural net that can represent the examples with all the censors and from which we can extract other cases which are not given in the examples list.

**Q. Why don't you transfer the censor production rules to standard production rules with negations to the required censors directly (why bother using a neural network)?:** The point is not a matter of transferring from one type of rules to another, the point is a matter of generalization. Moreover, whenever time limits to check all the censors or the average time to check a censor is changed, we only need to train the neural network.

## THE PROPOSED SCHEMES

To solve our problem, we use two different techniques. The first technique is the modified backpropagation and the second technique is applying backpropagation followed by special proposed algorithm. We shall call the first technique the modified BP and the second technique the hybrid approach.

**Modified BP:** In this method we use the ordinary backpropagation with a little modifications.

- Provide the system with the Number of censors in the System (NS).
- Provide the system with the importance of each Censor ($CI_i$).
- Provide the system with the Average time for checking any censor(AV).
- Provide the system with b and c values.
- Calculate the weights of the censors WVI,WI and WL.
- Randomly generate weights for all other remaining links.
- For each input condition or censor that applies for every trained rule has the value 1, otherwise the value 0. Similarly, the output that applies for the given input has the value 1, otherwise, 0.
- Apply backpropagation algorithm until we get the net outputs
- Keep calculating the error and adjusting the weights based on the backpropagation until we reach the input layer neurons. For input neurons, apply only the adjustment for the condition weights and keep the weights of the censors unchanged.
- GO to step 7 until all the assumed outputs (or mostly) are achieved(i.e., the error is very small).

In the previous algorithm, we get a net that can produce all the rules with full consideration of all the censors. One thing is to be noticed is that the censors have weights in which each reflects the checking time for the corresponding censor. This time is based on the average time for checking the censor and each censor's degree of importance. This time will be very useful in rule extracting in the next phase.

**Hybrid Approach:** We call this method hybrid approach because it uses two techniques to achieve the goal. The first is the ordinary backpropagation and the second is an algorithm which we develop to adjust the weights until we get the proper weights for the censors. This method can be summarized as below:

**Stage1:** Input : as explained before, it is a table representing the conditions and censors as inputs and the rule actions as outputs. For every provided input has value1,otherwise has value 0. Similarly, the corresponding output of the input has value 1, otherwise 0.
Process : Apply the backpropagation algorithm.
Output : A neural net that represents the rules that have trained for. All the weights of the net are just weights and do not reflect anything even those which correspond to censors.

**Stage 2:** In this stage we try to adjust the censor weights until they reflect the supposed time. This adjustment should also take place in the links that connect input conditions to hidden neurons and the links that connect the hidden layer with the output layer. This is done because we need to keep the new net produces the same output of the old net.
Input : The net obtained from stage1
Process : This is an algorithm to adjust the weights of the net until we reach with the weights of the censors to represent the assumed time for each (WVI,WI,WL). The algorithm is as below :

- Provide the system with Number of censors in the System (NS).
- Provide the system with the importance of each Censor ($CI_i$).
- Provide the system with the Average time for checking any censor(AV).
- Provide the system with b and c values.
- Calculate the weights of the censors WVI,WI and WL.

- For every censor, the achieved weight is either WVI,WI or WL based on the importance of the censor and the average time for checking the censor. We shall call the achieved weight for each censor AW.
- Define m with a small value.
- We define three values for g, $c_1$,$c_2$ and $c_3$ where $c_1 < c_2 < c_3$. $c_1$ corresponds to censor with very important degree of importance, $c_2$ corresponds to censor with important degree of importance, $c_3$ corresponds to censor with les important degree of importance.
- We define three values for x, $b_1$,$b_2$ and $b_3$ where $b_1 > b_2 > b_3$. $b_1$ corresponds to censor with very important degree of importance, $b_2$ corresponds to censor with important degree of importance, $b_3$ corresponds to censor with less important degree of importance.
- .Repeat steps 11 – 14 for n number of iterations
- totalerror = 0
- For each censor weight (say CW) (the weight of the link connects the input censor to hidden layers ) do the following:

```
error = 0
if ((CW – AW) is positive) then
if (CW-AW <= m) then
CW = (CW+AW) / 2
error = CW-AW
else /* in case CW-AW > m */
        CW = CW – g /* we reduce lesser value for higher importance censor */
        error = g
else /* This means CW < AW */
        if (CW is positive) then
        if(AW – CW <= m) then
CW = (CW+AW)/2
error = AW-CW
else /* AW – CW > m */
CW = CW + x
error = x
else /* CW is negative */
CW = CW + x
error = x
```

- ./* This part is to modify the links between hidden layers and output layer */ For each link connecting the hidden layer to output layer do the following (let us call the link's weight is WO) :

```
If (WO) is positive then
 WO = WO – error/ Numout /* Numout is the number of neurons in the output layer */
else  WO = WO + error/Nnmout
```

- totlerror = totalerror + error /* This part is to modify the weights of the conditions, these are the weights for the links that connect conditions with the hidden layers, say we call this weight WW */
- We compute the sum of all the weights related to conditions

$$\text{Sumw} \sum_{i=1}^{y} (WW)$$ , where y is the number of conditions

- if (sum w is positive) then

```
WW = WW – (error/sumw) * WW
else
 WW = WW + (error/sumw) * WW
```

**Output:** The output of this stage is a net that can produce correct conclusions with censors weights that can reflect the checking time for each.

## RULE EXTRACTION

After obtaining the net produced by the modified BP or hybrid approach, we can extract the rules that can consider censors that will be checked within time limits. Those censors negated will be added to the rule as main conditions. The summery of the proposed algorithm to extract thr rules is as below:

- Compute the number of censors that can be checked within time constraints
  NCC = TL/AV
- For all the conditions with value 1 are considered as main condition connected with AND relation.
- Sort all the censors according to their weights,say the list is L.
- For (i=0;i<NCC;i++) /* this is as the c language loop form */

- • Choose the censor with the highest weight from the sorted list, say censor co.
- • Consider the censor co to be in the rule as a condition with negation
- • Remove the censor co from the list L.
- Consider the output with the highest value as the output of the rule.

## RESULTS

A part of Haddawy[14] 's estimator knowledge base for CPRs system is used to test our proposed system. The knowledge base used is concerned with equipments necessary for heating system. Any heating system may consist of seven main components, room thermostat, duct sensor, pipe sensor, valve, panel controller, heating coil and fan. The system is to learn selecting the proper equipment. In present experiment we limited ourselves with valve equipment .In present valve equipment, we have V1,V2,V3,V4,V5,V6 and V7 type of valves. Each type is selected based on certain given situations such as type of power (electrical or nuclear ) ,type of control, pressure range and accuracy. Some of the censors are type of durability and speed of delivery and safety factor.

We have 11 rules in which some contain censors and others do not. Moreover, some rules produce more general output than others. To clarify these two points, we present the following two rules :

- IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELECTHEN ASK-FOR-EQUIP-CODE: V1-V2-V3 UNLESS
- IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC AND PRESSURE:3-7 AND ACCURACY:2 THEN ASK-FOR-EQUIP-CODE:V2 UNLESS DEL:SLOW, DUR:MED, SAFETY:LOW

The first rule says, if its condition is satisfied, then ask for valves of type V1, V2 or V3. If the condition of the second rule is satisfied, the ask for valves of type V2. We notice that the first rule is more generic than the second rule. Moreover, the UNLESS clause of the first rule is empty while the second has three censors.

**Sample run:** In our implemented system we have 9 conditions and 6 censors. Regardless of the conditions, we here care about the censors to show how the censors are chosen to be checked. The set of censors we have and the assumed degree of importance for each censor is given below

| Number | Censor | Degree of importance |
|---|---|---|
| 1 | DUR:HIGH | 1 |
| 2 | DEL:FAST | 2 |
| 3 | DEL:SLOW | 3 |
| 4 | DUR:MED | 1 |
| 5 | DUR:LOW | 2 |
| 6 | SAFTEY:LOW | 3 |

We notice that various values of DUR (HIGH,MED,LOW) may have different degree of importance.

We assume further that the TL is 6 time units and the values for b and c are 0.2 and 0.6 respectively and the average time for checking a censor is 3 time units. This means a maximum of two censors will be considered based on each's weight. We should finally get a net which produces correct outputs with weights for the censors 1 and 3 equal to 3(about to 3) and censors 2 and 4 equal to 2.8 and censors 3 and 6 equal to 2.4. In general the idea of assigning the full AV to those censors which have very important degree of importance is that many of them may be considered in one rule. For example, if in our case two of the censors with very important degree of importance are in one rule, then these two rules will be considered because their sum weight is 6 which is TL.

In both the approaches modified BP and hybrid approach, we got similar results in terms of the produced rules (which is our goal) but not in terms of the weights of the links. For sure, the weights for censors are almost the same. We shall discuss some variations in the two approaches in the next section. To see the system performance, we shall consider some of the rules given at the beginning to the system and the obtained rules after extracting them from the trained net.

Some of the training data considering all the censors will form the following rules

**RULE 1:** IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC
THEN EQUIP-CODE: V1-V2-V3
UNLESS

**RULE 2:** IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC AND PRESSURE:3-7 AND ACCURACY:2
THEN EQUIP-CODE:V2
UNLESS DEL:SLOW, DUR:MED, SAFETY:LOW

**RULE 3:** IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC AND PRESSURE:1-5 AND ACCURACY:2
THEN EQUIP-CODE:V2
UNLESS DUR:HIGH, DEL:FAST, SAFETY:LOW

THE EXTRACTED RULES BY THE SYSTEM CORRESPONDING TO THE ABOVE GIVEN SET ARE SHOWN BELOW :

**RULE 1:** IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC
THEN EQUIP-CODE: V1-V2-V3
UNLESS

**RULE 2:**IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC AND PRESSURE:3-7 AND ACCURACY:2 AND NOT (DUR:HIGH) AND NOT (DEL:SLOW)
THEN EQUIP-CODE:V2

**RULE 3:** IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC AND PRESSURE:1-5 AND ACCURACY:2 AND NOT(DUR:HIGH) AND NOT (DEL:FAST)
THEN EQUIP-CODE:V2

It is clear from the extracted rules that the system considers the censors with higher importance to be as main conditions for the rule. It is to be noticed that the extracted rules do not contain UNLESS clauses which can still be considered for more checking if some time is left due to some computation error in our algorithm. Therefore, it simple to extract rule 3 as following

**RULE 3:** IF EQUIP:VALVE AND CONTROL:TEMP AND POWER:ELEC AND PRESSURE:1-5 AND ACCURACY:2 AND NOT(DUR:HIGH) AND NOT (DEL:FAST)
THEN EQUIP-CODE:V2
UNLESS SAFTEY: LOW

## REMARKS

One of the important observation throughout several runs is that, in hybrid approach the output values of the general concepts comparing to more specific ones are very close (but less) in cases of specific concept inputs. For example in case of the input for rule 2, we may get an output value for it as (which is the highest) 0.98, whereas the output value corresponding to rule 1 (with inputs for

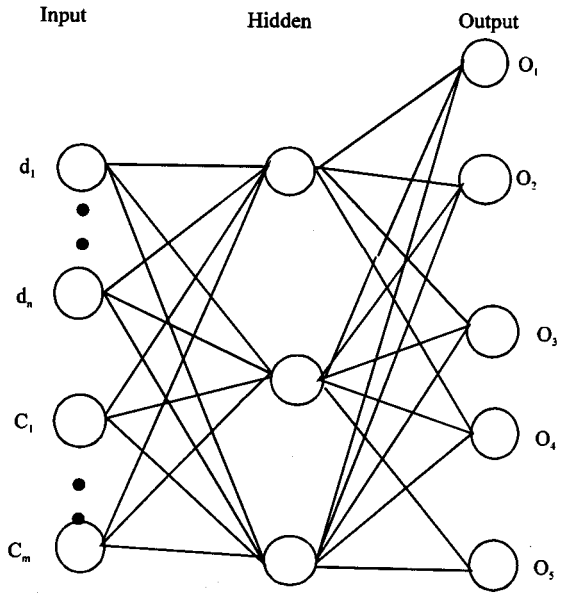


Fig. 1: The neural net structure. $d_i$ is the ith condition and $c_i$ is the ith censor. O $_i$s the ith output indicating the ithe rule action

rule 2) may be 0.96. In modified BP, the value of the correct output is 1 and wrong output is 0. We mean given the same above case for rule 1, rule1 s output value is 1 and output value for rule 2 is 0. The hybrid approach will be good in cases we want to know the more general concepts. This is also could be useful in HCPRs and GRSs . A third experiment has been done by using genetic algorithms instead of our proposed algorithm (stage 2 in hybrid approach). The required results obtained after very long time and too many generations (Fig. 1).

## CONCLUSIONS

Training a neural net to represent censor production rules and a method to extract the rules in terms of time constraints are presented. For training purpose, we used two methods, one is based on backpropagation with a little modifications called (modified BP). The second is based on a hybrid approach of backpropagation and a proposed algorithm. The two methods take into account the importance of each censor and the average time for checking each censor. The censor weights should finally reflect the time that will be considered for checking each censor. Based on the trained net, we can extract rules that can consider the censors based on their weights and the response time. The extracted rules will not have censors and would include negated censors as conditions. The included negated censors are those with higher

importance and which can be tested within time limits. Implementations show that the system in both the methods, modified BP and the hybrid approach give the same results. An observation has been noticed using the hybrid approach in which we may be able to know general as well as specific concepts which can t be with modified BP. Some of the future directions are (1) modifying the proposed algorithm to serve HCPRs and GRSs. (2)Using a hybrid approach of neuroevolutions to explore more possibilities for real time systems (this part is currently under research).

## REFERENCES


1. Fan, J., R. Lau and R.Miikkulainen, 2003. Utilizing domain knowledge in neuroevolution, Proceedings of the Twentieth Inter. Conf. On Machine Learning (ICML-03) Washington,Dc,
2. Gomez, F. and R. Miikkulainen, 2002. Learning robust nonlinear control with neuroevolution (Technical Report AI01-292). The University of Texas at Austin
3. Towell, G. and J. Shavlik, 1991. Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. Advances in Neural Information Processing Systems,Vol.4,Denver,CO Morgan Kaufmann, pp: 977-984.
4. Towell, G. and J. Shavlik, 1994. Knowledge based artificial neural networks, Artificial Intelligence, 70: 119-165.
5. Wermter, S., 1997. Hybrid approaches to neural network based language processing (Technical Report TR-97-030). UC-Berkeley,Berkeley, CA.
6. Zhou, Z., Y. Jiang and S. Chen, 2003. Extracting symbolic rules from trained neural network ensembles. AI Communications, 161: 1-14.
7. Zhou, Z., Y. Jiang and S. Chen, 2001. A general Neural framework for classification rule mining, Intl. J. Comp. Sys. Signals, 1: 154-168.
8. Zhou, Z. and S. Chen, 2002. Hybrid decision tree, Knowledge-based Systems, 15: 8.
9. Michalski, R. and P. Winston, 1986. Variable precision logic. Artificial Intelligence, 29: 121-145.
10. Bharadwaj, K. and N. Jain, 1992. Hierarchical Censored Production Rules (HCPR) system, Data and Knowledge. Engineering, 8: 19-34.
11. Hewahi, N.M., 2002. A general rule structure, J. Inform. Software Technol, 44: 451-457.
12. Compton, P. and D. Richards, 1998. Taking up the situated cognition challenge with ripple down rules Intl. J. Human. Computer. Studies, Special Issue on Situated Cognition 49: 895-926.
13. Hewahi, N.M., 1998. Real Time Variable Precision Logic Systems, F. Flavio (Ed.) Lecture notes in artificial Intelligence, 1515: 201-208.
14. Haddawy, P., 1987. A variable precision logic inference system employing the Dempster-Shafer uncertainty calculus. MS. Thesis (UILU-ENG-86-1777), University of Illinois at Urbana champaign.