
Proceedings Paper:
Rudd-Orthner, R. and Mihaylova, L. (2020) An algebraic expert system with neural network concepts for cyber, big data and data migration. In: Proceedings of 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT). 19th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2019), 10-12 Dec 2019, Ajman, United Arab Emirates. IEEE . ISBN 9781728153421

https://doi.org/10.1109/isspit47144.2019.9001880

An Algebraic Expert System *with* Neural Network Concepts *for* Cyber, Big Data and Data Migration

Richard N M Rudd-Orthner[1,2]
[1] *Department of Automatic Control and Systems Engineering*
University of Sheffield, Sheffield, UK,
RNMRudd-Orthner1@sheffield.ac.uk
[2] MASS KSA (a Cohort plc company), Riyadh, KSA.
rruddorthner@mass.co.uk

Lyudmilla Mihaylova
Department of Automatic Control and Systems Engineering
University of Sheffield
Sheffield, UK.
L.S.Mihaylova@sheffield.ac.uk

Abstract—This paper describes a machine assistance approach to grading decisions for values that might be missing or need validation, using a mathematical algebraic form of an Expert System, instead of the traditional textual or logic forms and builds a neural network computational graph structure. This Experts System approach is also structured into a neural network like format of: input, hidden and output layers that provide a structured approach to the knowledge-base organization, this provides a useful abstraction for reuse for data migration applications in big data, Cyber and relational databases. The approach is further enhanced with a Bayesian probability tree approach to grade the confidences of value probabilities, instead of the traditional grading of the rule probabilities, and estimates the most probable value in light of all evidence presented. This is ground work for a Machine Learning (ML) experts system approach in a form that is closer to a Neural Network node structure.

Keywords—Artificial Intelligence; Knowledgebase; Expert Systems; Neural Network; Information Assurance.

I. INTRODUCTION

Intelligent reasoning is a high level human pursuit of truth which arguably requires developed abilities for objective viewpoint reasoning and creativity, and those mix of abilities are perhaps not naturally found in every person in every circumstance. Also human reasoning requires to be able to deal with partial, inaccurate and uncertain information to produce higher order knowledge and understanding. This is a challenge for Computational techniques and Neural Network approaches have been used in this area but arguably have application challenges for validation when training datasets are incomplete, do not exist or would of contain unexpected values [1]. Also expert systems are a human assistance for consistent decision making [2], although one of the greatest challenge is not the actual content of the optional decision permutations, but is the grading of confidence in those decision permutations [3]. Another approach is a paper by Melen et al [4] that looks at machine learning for an expert system using a rule base and a Bayesian network for rule selection. This paper's approach is distinct from that and is part of a neural network approach using formula extraction as a compliment to rule extraction where the early work was published recently [5]. Outside of computational methods, often the grading of confidence is very difficult, and is often a more intensive activity then the understanding of the decision permutation itself, and in some cases, outside of computational machine assistance it is not invested in or relies on a voting system of human opinions [6]. However, within computational methods, approaches have used data mining within textual and imagery databases and can bring benefits to validation [7]. Traditionally Expert System approaches are largely textual hierarchies of rules in either forward chaining as an Explainable Expert System (ESS) or backward chaining executions to arrive at a decision or the reasoning evidence for a decision as in the Reconstructive Explainer (Rex) [8]. Sometimes also with scored multi-hypothesis alternatives or concepts of knowledge type [8, 9]. Approaches with statistical probabilities have been proposed in safety critical applications like medical [10] and military Data Fusion [11] and relate to a probability of decision tree rules with a single input instance of each input. However, this paper presents an algebraic form of an Expert System where a differing number of alternative input values will be combined, to form a perfected value and confidence from all of the possible evidence values of those inputs, rather than a probability for a decision tree rule being likely as in other approaches. That is to say the alternative decision permutations convergence and distribution will modify the expert systems result, both in-terms of the missing values and also in resultant confidences for those values. The growing development pace in *Artificial Intelligence (AI)* and *Cyber* poses the deeper reasoning approaches by safety critical systems to take actions based on observations with greater *Information Assurance (IA)* [12]. That adaption is also moving precedence for more *"on the fly"* (or Online) adaption with safety critical outcomes. Rationally, machine reasoning has the greatest value when the information is uncertain, inaccurate, unclear, conflicting or incomplete and approaches have incorporated Bayesian networks [13].

II. PAPER STRUCTURE

Section one is the introduction of the application area, and the novelty of modernizing the expert system approach to be mixed with Neural Network concepts, part of that novelty is that the expert system approach is grading a value rather than grading a rule, Section two is the paper structure. In section three is the mapping of computational Neural Network approaches and the human intelligence concepts to be combined into this Expert System data approach. Section four is the input representation discussion of the benefits of a language. Section five discusses a multi hypothesis approach for an algebraic form of a expert

system. Section six is an knowledgebase language syntax and organization. Section seven is the data structure used to represent the algebraic form as a computational graph. Section eight is the construction of the computational graph, and section nine the evaluation processing. Section ten is the estimation of a final value based on deduced confidences. Section eleven structures into decision trees by guard equations to form valid populations of hypothesis. Sections twelve and thirteen are: an illustrative example using a prototype and the summary and conclusion.

III. Structure Of Data

In human terms approaches to resolving this can be intellectually deductive and inductive as per Aristotle's and Sir Frances Bacon's reasoning [14], but may be applied to computational techniques like Neural Networks and Expert Systems. Machine reasoning can be seen as the transformation of observations and with some knowledge can create understanding. Although it may be that there is some uncertainty in those observations that may create more than one version of an understanding, as possible knowledge based permutations, based on different knowledge rules when applied to those observations. When considering these transformations, data can be represented in three forms as a structured approach that bridges between Intelligence of humans and machine learning approaches and based on the Intelligence approach as a Life Cycle [15], three forms of data are described below:

- **Raw Data**: The output from the Intelligence Life Cycle stage "Collection", and is the input to the "Processing" stage. In Neural Networks this can be seen as the pre-processing and Input layer, as observations from a collection activity are presented as the evidence in the input layer.
- **Information**: The output of the Intelligence Life Cycle "Processing" stage. In Neural Networks this can be the output of the hidden layers. It converted Raw Data into Information through a "Processing" activity prior to categorization in the output layer.
- **Intelligence**: In the Intelligence Life Cycle this is the output from the "Analysis and Production" stage. In Neural Networks this is the Output layer categorization and converts Information into Intelligence.

A. Definitions Knowledge-Base File Compartment

These are rules that contain quantities and conventions that may exist to allow data to be understood generically in all knowledgebase compartments, and in this approach these are rules that contain global quantity conversions and standard value definitions.

B. Raw Data Knowledge-Base File Compartment

Raw data is often formatted in a source unique way, and may contain both relevant and irrelevant information combined. It is processed so it can be used purposefully were the relevance is increased. In a neural network approach this is like the pre-processing and input layer, in this Expert System mixed approach this is a source conversion layer to a generic form, where the rules can use scientific unit quantities defined in the definitions such that the subsequent hidden layer or layers can be re-used.

C. Information Knowledge-Base File Compartment

Information is catalogued raw data that prioritizes a collected raw dataset into useful information and is the processed form of the raw data extracts with a standard understood semantic and syntax. In a neural network approach this is like the hidden layers, in this Expert System mixed approach this is where the generic reusable rules can be fused (or "converged"), calculated and graded prior to final representation as Intelligence but provides reusable libraries of knowledge transformation.

D. Intelligence Knowledge-Base File Compartment

Intelligence is where the contextual understanding is combined to the information (processed raw data) to answer a specific question in exploitation, and it takes a knowledge context and understanding to answer a question. In a neural network approach this is like the output categorization layer. In this Expert System mixed approach it is the re-representation in a form that would be understood by the user, and has applications to database data migration.

These structured data compartments presented here forms a kind of bridge between human organizational use for analysis techniques used in the Intelligence life cycle, computational approaches like neural networks and the structure proposed in this Experts System approach. However, within these organizational knowledgebase compartments there is also a requirement for a knowledge definition representation that is consistent between them.

IV. The Input Representation

Representing knowledge creates the need for a representation that is flexible to not compromise the semantics of the knowledge by making it fit a standard model unless it is suitably flexible and easy to use. That model needs to represent the knowledge with low compromise and with low abstraction, and resultant knowledge of the understanding that can be evaluated for confidence. Traditionally the Relational Database Management System (RDBMS) and eXtensible Markup Language (XML) are forms that involve a configurable complexity to provide enough flexibility in a fixed data structure, but has a drawback of representing a data centric view, rather than logic models like are used with machine operations and in algorithms. However, the database and XML representations have high syntactic value for raw data, but may have in conversion lost some semantics in conversion. In fact in may be argued that there is a need for a language syntax form to capture both data values and machine logic operations together. Furthermore, the conversions between Raw Data, Information and Intelligence may change confidences or combine sources where as the Raw form may be more well understood. So these sources and confidences require to be tracked from the Raw Data to the intelligence conversions in terms of security source caveats and confidence weightings. Such that the output can report and express security source caveats and confidences as metadata in the resultant output estimations.

V. The Multiple Hypothesis Approach

Considering that a system taking observations of raw data and using rules of established data conversions to calculate possible missing data (i.e. values or understandings that were not observed), Then this could be considered as a

multiple hypothesis approach when there are alternative valid rules. Where those multiple hypothesis are captured as rules and alternative data values from separate sources, then all the rule permutations may be applied to all source data values from each of the sources to calculate all the possible combinations of values. Then a confidence may be derived from how well the multiple hypothesis rule and value combinations results numerically match together. These can also be scaled by the source confidence and the population of results, such that the confidences are commutable with other rules sets that have a different number of rule hypothesis. To capture both the knowledge conversions as rules and possible observation values, a consideration of a knowledgebase representation can be considered in a mathematical algebraic form.

VI. Knowledge Representation

A representation is required to capture knowledge in a form where conversion can be evaluated. The illustrated form captures mathematical relationships using an algebraic equation form such that different known data (atoms) can be fed into the knowledgebase rules (axioms) and will include guard equations, security caveats and rule confidence weightings. To parse these axioms and atoms, a language format with low abstraction in the representation is required and has the following formal Backus Nour Form (BNF) syntax expression (see Figure 1):

```
<KnowledgeBase> ::=  <AxiomAtom> ";" [<KnowledgeBase>] ;
<AxiomAtom>     ::= <Symbol> "=" <Expression>
                    [ "When" <Expression> ]
                    [ "Security" <Integer> ]
                    [ "Confidence" <Expression> ] ;
<Symbol>        ::= <Identifier> ;
<Literal>       ::= <Real> | <Integer> ;
<Expression>    ::= <Term> [ <Relation> <Expression> ] ;
<Relation>      ::= "<" | "<=" | "==" | ">=" | ">" | "!=" ;
<Term>          ::= <Factor> [ <Operator> <Expression> ] ;
<Operator>      ::= "+" | "-" ;
<Factor>        ::= <Quantity> [ "*" | "/" <Expression> ] ;
<Quantity>      ::= <Value>  [ "&&" | "||" <Expression> ] ;
<Value>         ::= [ <Operator> ] <Parameter> ;
<Parameter>     ::=  <Symbol> | <Function> | <Literal> |
                    "(" <Expression> ")" ;
<Function>      ::= <Symbol> "(" [ <ParameterList> ] ")" ;
<ParameterList> ::= <Expression> [ "," <ParameterList> ] ;
```

Fig 1 BNF Expression of Knowledgebase Language

The BNF expressions can be used with both: Atoms (observational facts) and Axioms (knowledge rules), and they are separated into different files to be parsed using the BNF, so their difference can be recorded in a symbol table.

VII. Algebraic Representation

To represent an algebra within a computer requires consideration of an abstraction to nodes with relationships that may form hierarchical structures much like a Neural Network Computational Graph. A node may result in a value like either: *Symbol* which is a reference to an *identifier* of a variable parameter, a *Literal* value which is a real or integer constant value, a *Function* with an associated *Parameter List.* A node can also be a machine operation like a *Operator*, *Factor, Term or Equals*. As nodes need to form hierarchical tree structures, they need tree pointers to form the links to related nodes, and as a basic mathematical operator has an action and two operands a binary tree structure seems appropriate. Although in the case of a function call it should not be limited to two parameters. So the tree pointers relate to a below level, and a below then same level ordering of nodes. As such the nodes show the required parameter list at the same level for a node, and at the below levels the relationships of connected chains of dependant operations for those parameters that were at the function's below and same level node's parameter list. In this way a computational graph is built into atomic operations in each related node and forms the relationship dependencies. This form of a binary tree of dependant computational chains maps onto the Reverse Polish Expression stack based execution model for evaluating results. As there may be many rules and many values each node needs to capture a list of values that will be computed, and so that values for rules and observational evidence can have differing confidences and security caveats. Each node needs to track these with the value list as they are combined in nodes during evaluation.

VIII. Building The Computational Graph

In terms of execution order, and using recursion on each node a depth first then breadth node order would compute each depended chain before each parameter is required and can be converted to a Reverse Polish Expression for evaluation, that is executing in turn from the leaf nodes upward to the root node. A fact value can be added as an assignment and will allow many fact values to combine alternative values. Each rule and fact tree can be added as a tree from the same level pointer of the root node, to compose a single tree for the whole knowledgebase. A computational graph structure is built in three parts: a symbol table, a hierarchical node structure and value list. Each node is added to a symbol table during the parsing. The Hierarchical tree points to the symbol table which has the initial values with their parse time known values, in the case of literal values it is added to the hierarchical node structure's literal node as a single value. Security caveats and confidences are added to the upper most node in the hierarchical node structure for that rule or fact. If the Security Caveat is not defined in the knowledgebase then it is assumed to be 0x00000000, and as it is a bit field this means *No Caveat*, these are the defaults for all other nodes. Also if the confidence is not defined in the knowledgebase it is assumed to be the value 1.0 (100%). Each node is inserted above the last node but where an operator has an operator precedence then the node is added below so it will execute first but that is dependent on the relative operator in the last node. Brackets are implemented as functions and functions are added below, with their parameter list computational graph structures added on the same level of the function node's below node. Each of the four data compartments (Defines, Input, Hidden and Output layers) are loaded in order, building the complete knowledgebase, but their structure allows reuse.

IX. Computational Graph Evaluation

On evaluation the hierarchical nodes are computed, but depending on the node type the machine operation is different:

Literal values are taken from the symbol table and loaded in to the hierarchical node's value list for further computation by nodes connected to them, as in Figure 2.

$$Value = SymbolTable.Value$$
$$Confidence = SymbolTable.Confidence\ (default\ 1.0)$$
$$Security = SymbolTable.Security\ (default\ 0x00000000)$$

Fig 2 Literal Type Node Computation

Operators take the list of values from the child node *(Below)* and the child node's same level node *(Below.Same)*, the operator defined in the semantic field is used to calculate all resultant values mixes, and are added to the value list of that operator node. The confidence is calculated from multiplying each of the operand values' confidences. The security caveats are tracked by taking the BINARY OR of the security caveats from both operands for each individual value calculated (see Figure 3).

$$Value = Below.Value\ \langle \boldsymbol{Operator} \rangle\ Below.Same.Value$$
$$Confidence = Below.Confidence \times Below.Same.Confidence$$
$$Security = Below.Security\ \boldsymbol{OR}\ Below.Same.Security$$

Fig 3 Operator Type Node Computation

Functions are the same as operators although the number of child same level nodes are dependent on the number of function parameters used.

Variables get all the possible values for every matching variable name in the tree, then sets the confidences from the product of the minimum ratio between each other value combinations, divided by the population and scaled by this variable nodes own confidence if it was set in the parse time, for that knowledgebase rule's root node. Security Caveats are the security caveats from the variable's search values as in Figure 4.

$$Values = Search.Values$$
$$Diff = min\left(\frac{Search.Confidence(x)}{Search.Confidence(\mathrm{y})}, \frac{Search.Confidence(x)}{Search.Confidence(\mathrm{x})}\right)$$
$$Confidence = \frac{prod(Diff\{0..\})}{Population \cdot OwnNodeConfScaling}$$
$$Security = Search.Security$$

Fig 4 Variable Type Node Computation

The nodes are processed in a recursive fashion until all nodes are computed, the computation does imply that all input values need to be calculated first for the tree searches to have all the values loaded, but the compartmentalized knowledgebase file structure should support this.

X. Estimation of Values

To make an estimate of a perfected value and also to be consistent with probability trees the addition operation is used in a histogram of the values, with the cumulative summation of confidence for each value in the respective bin. So a *centre of gravity* can be taken to arrive at a perfected value and a perfected confidence for that value. The variable node in the hierarchical tree provided the scaling of the confidences to normalize the confidence consistently such that they are commutable. A histogram is compiled from: a dataset length *(NoOfValues*), the max and min values in that dataset that are *MaxValue* and *MinValue,* see Figure 5.

$$ValueRes = \frac{MaxValue - MinValue}{max(min(NoOfValues, MaxHistSize) - 1, 1)}$$
$$d = \left\lceil \left(\frac{MaxValue - MinValue}{ValueRes} \right) \right\rceil$$
$$CulSum = \sum (\{0..d\} \cdot ValueRes + MinValue) * hist[0..d])$$
$$ValueSum = \sum (\{0..d\} \cdot ValueRes + MinValue)$$
$$HistSum = \sum hist\{0..d\} \qquad WeightValue = \frac{CulSum}{HistSum}$$
$$Certainty = \left(\frac{CulSum}{ConfSum \cdot HistSum} \right) \cdot 100$$

Fig 5 Centre of Gravity from Histogram

Although a histogram has a bin resolution *(ValueRes)* the *centre of gravity* process will provide a sub bin resolution. The *centre of gravity* process provides a perfected value that takes the distribution and the relative confidences in that distribution into account. So that taking all the processing into account the final value is a perfected value based on all the confidences from every valid rule permutation, because the confidences were set based on how many alternative answers there where and how close they matched in value from separate rule or value permutation routes. That is to say they measure how well matched the resultant values are that came from alternative rules or known observations as a match to known understanding in the knowledgebase. This is true so long as all the rules that are accessible are all valid for the conditional circumstance, so a guard equation is provided to each rule to control the population of answers to a set that are valid.

XI. Population Validity Decision Trees

So that the population of values used in the knowledge base are valid a guarded equation can be used to wire out rules under conditions set in the rules or in input values, in much the same way as existing Expert Systems structure their knowledge in the knowledgebase to form decision trees. The guard equation can be activated via the *"When"* clause and include or exclude rules to control the valid population of alternative answers under a condition deduced from the rules in the knowledgebase. If a value is set to *unknown* rather than a specific value, then the population of rules can include results for all conditions or be limited to some values, making the estimated perfected value and its' confidence reflective of only the valid answers.

XII. An Illustrative Example

Using a software prototype, an illustrative example: *SpeedOfLight = 299792458.0;* which is an assignment of a value is explained. In this example if a definition of the *speed of light* was used then the symbol table would contain the atom variable, the equation operator and the literal value for the assignment as this form is representing both data values and machine operations within the symbol table. The content of the symbol table is in Table 1:

Symbol Name	Type	Value
VAR00|SpeedOfLight	Var	*NotSetYet*

Equal01	=	Equal	*NotSetYet*
VALUE02	float	Literal	299782458.0

Table 1 Symbol Table Example

The symbols within the symbol table (Table 1) are loaded in as nodes (aNode) in an ordered binary tree as a fast reference indexing system, see graphical depiction in Figure 6:

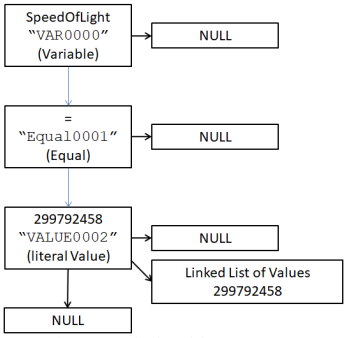

Fig 6 Symbol Table Structure

Each node *(aNode)* within the symbol table contains the information in Table 2 as a data structure for each node:

Structure Item	Structure Item Description
Ref Name	Unique Symbol name reference.
Type	Symbol node type.
Semantic	Semantic function indicator.
Value Ptr	Initial value in the linked list.
Security	Security tracking value loaded.
Confidence	Graded confidence value loaded.
Left, Right	aNode Tree pointers.

Table 2 Symbol Table Node Structure (aNode)

As there are multiple hypothesis values a link list of values provides a variable number of values at each node with a backward pointing reference of origin to collate security caveats and combine confidences inherited. The value linked list node structure is as in Table 3:

Structure Item	Structure Item Description
The Value	Value of this permutation.
Confidence	Confidence of this Value
Security	Security Caveats Tracking
Origin 1	Where value was calculated from.
Origin 2	Where value was calculated from.
Next	Pointer to the Next value in the linked list

Table 3 Value Linked List Node Structure

Each symbol table node (aNode) can be referenced from a hierarchical tree node *(aHierarchicalNode)*. The hierarchical tree is a binary tree of references into the symbol table tree and is an algebraic hierarchical structure that holds the computational graph structure, which references the: values, equals, operators and functions that are constructed into a *Reverse Polish* form in their hierarchical structure. The hierarchical tree node data is in Table 4 and provides the reference to the symbol table entry through the member pointer.

Structure Item	Structure Item Description
Value List Ptr	Calculated values linked list.
Member Ptr	Pointer to Symbol table (aNode).
Guard Hierarchal	Guard Equation aHierarchicalNode
Below, SameLevel	aHierarchicalNode tree pointers.

Table 4 Hierarchical Node Structure (aHierarchicalNode)

The guard equation is also referenced into a separate hierarchical tree structure for the ultimate evaluation of whether to include or exclude the rule structures hypothesis permutation. This allows rule and value structures to be included or excluded based on deduced values using logic terms to form decision trees and enforce that the population of hypothesis rule permutations is controlled and valid. Using the earlier example the hierarchical tree node relationships for that form are as in Figure 7 and are mapped to the original symbol table tree form:

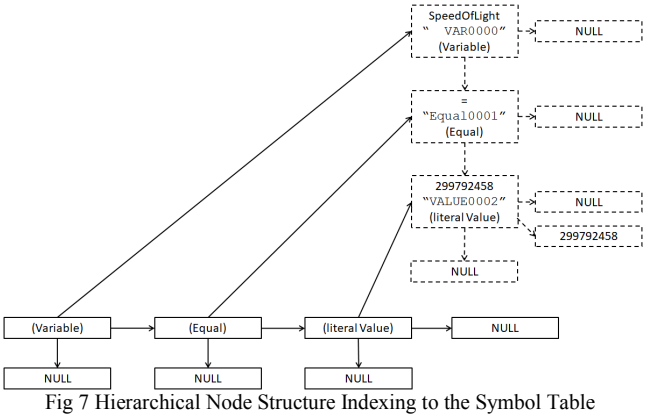

Fig 7 Hierarchical Node Structure Indexing to the Symbol Table

Note that the hierarchical tree nodes forms a reverse Polish expression that is a useful form that allow stack-based calculations. But this can be a calculation using operators and follow a form of values and operators. Where the algorithm will load the value onto the stack with the current stack value and then performs that operation on the stack it leaves the result on the stack for the next operation. This sequence is repeated until all operator and values are used leaving the final value. To support BODMAS (Brackets, Order, Divide, Multiply, Add and Subtract) the operator and hierarchical tree need to be populated in a form that builds the right hierarchical tree. For example *a = 1+2/3*, the hierarchical tree of operators and functions observing BODMAS are converted into a hierarchical tree in Figure 8 in a graphical form:

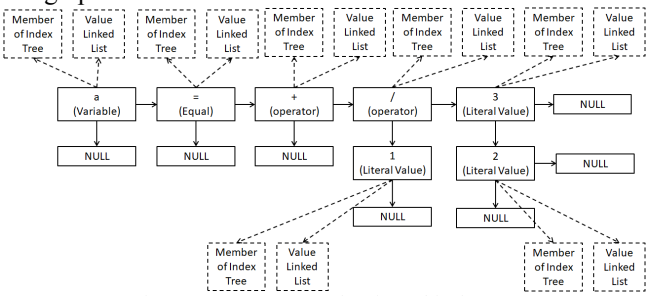

Fig 8 BODMAS Example Hierarchical Tree

This hierarchical tree can be evaluated from the leaf node on the right upwards towards the root node on the left. A depth, then breadth transversal provides the first value to be pushed on the stack and is the literal value 2, then literal value 3 is pushed and applies the operator '/' to calculate the value 0.66667. Then the literal value 1 is pushed onto the stack and the operator '+' is applied calculating the value 1.66667 which is assigned to the variable 'a', the Reverse Polish Expression that is formed is *2 3 / 1 + = a [1.66667]*. Adding nodes to the Hierarchical tree requires to be done in BODMAS compliant order while preserving a Reverse Polish Expression hierarchy. When an operator is added it

adds the denominator to the node below and the numerator at the same level node, this chain allows multiple same level nodes for operations like a function parameters lists that may have more than two subsequent chains as a parameter list. In the case if the Add/Subtract and Multiply/Divide the nodes are rearranged to place higher operator precedence below so they are evaluated before the addition. In the operator precedence Functions and Brackets are evaluated first and for this reason the brackets, power orders and Binary operators are represented as functions, as are also binary operators like AND and OR, which are used in the Guard Equations. The tree is evaluated from leave to root and forms the Reverse Polish Expression execution order.

XIII. Summary and Conclusions

It may be noted that the way that the hierarchical tree is populated may seem less obvious, as the two operand literal values: 2 and 3 are both below the operator "/" rather than one at the same level and the other one below. The reason for this is to allow function nodes to have more than two parameters and also allow the structure of the whole expression to be contained in one structure that is a forest of trees rather than as a number of individual trees. This also provides tree scope to be combined if a number of successive knowledgebase axiom rule-sets that are used separately are combined for successive deduction reasoning levels of logic like in a neural network layer with multiple hidden layers (as in Deep Learning). Also if there is "known data values" (as atom facts) then the axioms (rules) calculated values can be performed and confidences in each value set based on the nearness to the other values and the axiom estimate confidence as a ratio *(which is simply represent-able as a percentage to the user)*. This does mean the confidences are not confidences of truth but confidences of a fit to established axiom rules and other values, or rather a match to an established understanding captured in the knowledgebase. However, this also means that if the values are represented in a histogram them adding the confidences of equal values is permissible as it is consistent with a probability tree approach, thus the highest score will result from converging or matching answers. In conclusion the knowledgebase can be used to infill missing data for less observable data and also be used for data migration in support of databases. The layer structure of the knowledgebase is a similar concept to the neural network approach, and maps those layers to the input, hidden and output layers. Some rule exclusivity is introduced by the support of decision trees structures using guarded equations. Some rules can provide calculated results that may be invalid, so these rules inclusions can be controlled by and equation guard and these equation guards may separate rules to decide if a rule is valid given a control definition within the knowledgebase. This research has perhaps modernized a concept of Expert Systems with a Computational Graph approach to implement an algebraic form of a Expert System that is closer to a Neural Network concept. Forming a best answer in sight of the evidence using a confidence that is based in probability trees. This provides a forward chaining execution policy to combine with a neural network rule extraction method [5] in solving intractable problems.

XIV. References:

[1] Abdella, M. and Marwala, T. (2005). The use of genetic algorithms and neural networks to approximate missing data in database - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/1511574 [Accessed 25 May 2019].

[2] Bohanec, M. and Rajkovič, V. (2009). DEX: An Expert System Shell for Decision Support. [online] Citeseerx.ist.psu.edu. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.2037&rep=rep1&type=pdf [Accessed 25 May 2019].

[3] Voskoglou*, M. (2014). Measuring the Uncertainty of Human Reasoning. *American Journal of Applied Mathematics and Statistics*, [online] 2, pp.1-6. Available at: https://pdfs.semanticscholar.org/af03/f50acbabf84ff8ea221e83bcbc450573443c.pdf [Accessed 7 Aug. 2019].

[4] Melen, R., Sartori, F., & Grazioli, L. (2015). Modeling and understanding time-evolving scenarios. JOURNAL OF SYSTEMICS, CYBERNETICS AND INFORMATICS, 13(5), 62-67

[5] Rudd-Orthner, R. and Milhaylova, L. (2019). Numerical Discrimination of the Generalisation Model from Learnt Weights in Neural Networks. Annals of Emerging Technologies in Computing, [online] 3(4), pp.1-14. Available at: https://www.researchgate.net/publication/335023149_Numerical_discrimination_of_the_generalisation_model_from_learnt_weights_in_neural_networks.

[6] Johnson-Laird, P. (2010). *Mental models and human reasoning*. [online] Proceeding of the National Academy of Science. Available at: https://www.pnas.org/content/107/43/18243 [Accessed 7 Aug. 2019].

[7] Cooke, C., Santana, C., Morris, T., DeBraal, L., Ordonez, C., Omiecinski, E., Ezquerra, N. and Garcia, E. (2000). Validating expert system rule confidences using data mining of myocardial perfusion SPECT databases - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/898642 [Accessed 25 May 2019].

[8] Barzilay, R. and DeCristofaro, J. (1998). *A NEW APPROACH TO EXPERT SYSTEM EXPLANATIONS*. [online] Aclweb.org. Available at: https://www.aclweb.org/anthology/W98-1409 [Accessed 7 Aug. 2019].

[9] Khalak, Asif & Wemhoff, Eric. (2005). Multi-hypothesis estimation approach to diagnosis and prognosis of degrading systems. 3691 - 3701. 10.1109/AERO.2005.1559674.

[10] Spiegelhalter, D., Dawid, A., Lauritzen, S. and Cowell, R. (1993). [Bayesian Analysis in Expert Systems]: Rejoinder. Statistical Science, [online] 8(3), pp.277-283. Available at: https://www.researchgate.net/publication/257937651_Bayesian_Analysis_in_Expert_Systems_Disc_P247-283.

[11] Rauch, H. (1884). Probability Concepts For An Expert System Used For Data Fusion. [online] Pdfs.semanticscholar.org. Available at: https://pdfs.semanticscholar.org/9812/c3497dced433f59b66d4071835bd3c0f279d.pdf [Accessed 6 Sep. 2019].

[12] Varadaraju, R. (2011). *A Survey of Introducing Artificial Intelligence Into the Safety Critical System Software Design Process*. [online] Citeseerx.ist.psu.edu. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.1592&rep=rep1&type=pdf [Accessed 7 Aug. 2019].

[13] Wiegerinck, Wim & Kappen, Bert & Burgers, Willem. (2010). Bayesian Networks for Expert Systems: Theory and Practical Applications. 10.1007/978-3-642-11688-9_20.

[14] Cushing, J. (1998). Aristotle and Francis Bacon. In *Philosophical Concepts in Physics: The Historical Relation between Philosophy and Scientific Theories* (pp. 15-28). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139171106.004

[15] Recorded Future. (2018). *5 Phases of the Threat Intelligence Lifecycle*. [online] Available at: https://www.recordedfuture.com/threat-intelligence-lifecycle/ [Accessed 7 Aug. 2019].