A HARDWARE IMPLEMENTATION OF AN EXPERT SYSTEM SHELL BASED ON BELIEF REVISION CONCEPTS

Octavian Creț
Lecturer, Technical University of Cluj Napoca, Romania
Octavian.Cret@cs.utcluj.ro
Zoltan Baruch
Senior Lecturer, PhD, Technical University of Cluj Napoca, Romania
Zoltan.Baruch@cs.utcluj.ro
Kalman Pusztai
Professor, PhD, Technical University of Cluj Napoca, Romania
Kalman.Pusztai@cs.utcluj.ro
Calin Cenan
Professor, PhD, Technical University of Cluj Napoca, Romania
Calin.Cenan@cs.utcluj.ro

Abstract

We propose an Expert System Shell based on belief revision concepts who maintains the consistency of the knowledge base. In the first phase of this expert system shell we translate a classical rule-based system in an equivalent network representation where nodes are facts, and links stand for relationships. In the second phase we propagate any change in the uncertainty measures throughout the whole network.

This expert system shell was first implemented in software. The initial knowledge base is given in a text file, as a set of rules. The text file is parsed and a belief network is generated, having in its nodes the facts (propositions), linked by edges which contain the main operators: AND, OR, NOT, AGGREGATE and DET. The network also contains the ONE-OF operator.

This paper continues a previous research done by Cenan in [2], extending it to a hardware implementation. The hardware implementation achieves much better performance by exploiting the parallelism of triggering all the rules in the same time.

The system was implemented and simulated on a XC4005E FPGA device from Xilinx Corporation. The main advantage obtained is an increased speed of the implementation, due to the high level of parallelism achieved and to the intrinsic hardware features.

1. Introduction

The past years have witnessed a noticeable research effort towards a theory of reasoning under uncertainty. While probability theory was recently introduced in this area with the emergence of Bayesian belief network [3], the role of logic and symbolic representations will seemingly continue to be prominent. The monopoly of probability theory as a tool for modeling uncertainty has been challenged by alternative approaches such as belief functions. Current efforts seem to be directed towards the specification of a knowledge

representation framework that combines the merits of classical logic and Bayesian belief networks.

Solving this problem in a satisfactory way presupposes the next three requirements to be met and we consider that they are accepted by our approach.

1. The necessity of a clear distinction between factual evidence and generic knowledge
2. The need for representing partial ignorance in an unbiased way
3. The inference at work cannot be monotonic.

We try to represent uncertainty with a small set E = { E_i } of ordered linguistic variables, composed of nine elements; we will denote these nine elements by E_i, where i $\in\{1,2,\ldots,9\}$. The natural order induced among the variables holds true, i.e. E_1 stands for impossible; E_9 stands for certain.

$$E_i < E_j \Leftrightarrow i<j, \quad i, j \in \{1,2,\ldots,9\}$$

Uncertainty will be represented with an interval as a set of two parameters varying on the set E. This interval is formed by:

- support – the positive evidence for the assertion
- plausibility – the difference between the absolute certainty and the support of the negation of the assertion.

Support and plausibility are always independently updated, since they are defined as different kind of information associated to a proposition, separately acquired and conceptually unrelated.

Some criteria are provided, in order to measure the semantic relevance of the difference between belief intervals. Two distinct belief intervals are significantly different only if they belong to different belief states. The defined belief states partition in six areas the 9×9 table combining all the possible values for support and plausibility [1]:

S \ PL	**E_1… E_4**	**E_5… E_8**	**E_9**
E_1	D	U	
E_2… E_5	RD		
E_6… E_9	C	RB	B

B – believed
RB - rather believed than disbelieved
C – contradictory
U – unknown
RD – rather disbelieved than believed
D – disbelieved

Figure 1: *The values for support and plausibility*

2. Operators defined on belief intervals

To compute the belief interval associated to a compound expression, operators have been defined on belief intervals. We consider the notation |<exp>| as a shorthand for [s(exp), pl(exp)].

- **Negation**:
  The negation of a variable : $N(E_i) = E_9 - E_i = E_{9-(i-1)}$
  The negation operator for a belief interval: $NOT(|a|)=[N(p(a)), N(s(a))]=[s(-a), p(-a)]$
  The negation operator satisfies the involutive property: $NOT(NOT(|a|))=|a|$

- **Conjunction**:

$$AND(|a|, |b|) = \begin{cases} [E_1, E_9], \textbf{\textit{if}}\ |a| \in U, |b| \in C \\ [min(s(a),s(b)), min(p(a),p(b))], \textbf{\textit{otherwise}} \end{cases}$$

- **Disjunction**:

$$OR(|a|, |b|) = \begin{cases} [E_1, E_9], \textbf{\textit{if}}\ |a| \in U, |b| \in C \\ [max(s(a),s(b)), max(p(a),p(b))], \textbf{\textit{otherwise}} \end{cases}$$

- **Aggregation**: denoted by $\oplus$; aggregates the evidence coming from different sources to a single assertion. It can be used to combine evidence for an assertion which is present in the conclusions of more than one inference rule:

$$AGGR(|a_1|, \ldots, |a_n|) = [max(s(a_1), \ldots, s(a_n)), min(p(a_1), \ldots, p(a_n))]$$

where a_i stands for the evidence pertaining to a and coming from source i.

- **Detachment**: denoted by $\rightarrow$, propagates the evidence pertaining of an inference rule to its conclusions. The definition of the DET operator, with the belief interval pertaining to the premises of the rule |h| (hypothesis) and the strength of the rule $|h \rightarrow t|$ expressing the deduction is given below:

$$DET(|h|, |h \rightarrow t|) = \begin{cases} AND(|h|, |h \rightarrow t|), \textbf{\textit{if}}\ h \in B, |h| \in RB \\ [E_1, E_9], \textbf{\textit{otherwise}} \end{cases}$$

3. Translating rules to network representation

We represent knowledge by rules in a dependency network. In these networks there exist three types of nodes: *proposition*, *rule* and *operator*. Each rule node receives as input the belief intervals of its premises and produces as output the belief interval of its conclusion. Each operator node receives as input the belief intervals of its operandi and produces as output the belief interval of its result.

We assume that the propositions are predicate-value pairs. Some predicates are Boolean and we will consider only their positive form. Other predicates can take a single value in a larger set of alternative choices. This is a constraint that we have implemented by defining an operator "one-of" for belief intervals, having the same semantic as an "exclusive-or". This operator gives a result in the classes Believed (B) or Rather Believed (RB) if one and only one of its arguments belongs to the same class. The format rules is: rule name, used in the explanatory process, premise, a compound expression, conclusion, and strength, a measure of the uncertainty used as a belief interval for this rule.

Parsing a rule we must obtain a node with a detachment ($\rightarrow$) operator having the strength of the rule as a parameter. The two links of this node come from the antecedent of the rule and go to consequent of the rule. Both the antecedent and the consequent part of the rule could be nodes denoting simple propositions or nodes with operators taking place between propositions.

The initial knowledge base is presented as a set of rules; in our example, the format used to insert them in the network is the following.

R1: Strength 8
IF (Smokes-a-lot Yes)
THEN (Heart-risk Yes)
R2: Strength 8
IF (Stress Yes)
THEN (Heart-risk Yes)
R3: Strength 6
IF (Job Manager)
THEN (Stress Yes)
R4: Strength 6
IF (AND (NOT (Stress Yes)) (Temperament Shy))
THEN (NOT (Job Manager))
R5: Strength 8
IF (Temperament Vehement)
THEN (Heart-risk Yes)
R6: Strength 6
IF (Face-colour Ruby)
THEN (Temperament Vehement)
R7: Strength 7
IF (Temperament Sure)
THEN (NOT(Heart-risk Yes))

Figure 2: *The initial knowledge base*

Then, we obtain information and we supply the systems with evidences as external assumptions for proposition nodes. The observations could be expressed could be expressed in natural language as:

1) the patient has a ruby face
2) the patient seems to be shy
3) the patient is head of a department
4) there is a small chance that the patient is not stressed

The results are obtained in the following manner:

(1) (Assume (Face-colour Ruby) value (BeliefInterval (8 9)))
(Proposition (Temperament Shy) Evidence (1 9)U - (1 4)D)
(Proposition (Heart-risk Yes) Evidence (1 9)U - (6 9)B)
(Proposition (Temperament Vehement) Evidence (1 9)U - (6 9)B)
(Proposition (Temperament Sure) Evidence (1 9)U - (1 4)D)
(Proposition (Face-colour Ruby) Evidence (1 9)U - (8 9)B)

(2) (Assume (Temperament Shy) value (BeliefInterval (6 9)))
(Proposition (Temperament Shy) Evidence (1 4)D - (6 4)C)
(Proposition (Heart-risk Yes) Evidence (6 9)B - (1 9)U)
(Proposition (Temperament Vehement) Evidence (6 9)B - (6 4)C)

(3) (Assume (Job Manager) value (BeliefInterval (9 9)))
(Proposition (Stress Yes) Evidence (1 9)U - (6 9)B)
(Proposition (Heart-risk Yes) Evidence (1 9)U - (6 9)B)
(Proposition (Job Manager) Evidence (1 9)U - (9 9)B)

(4) (Assume (Stress Yes) value (BeliefInterval (1 6)))
(Proposition (Stress Yes) Evidence (6 9)B - (6 6)RB)
(Proposition (Heart-risk Yes) Evidence (6 9)B - (6 6)RB)

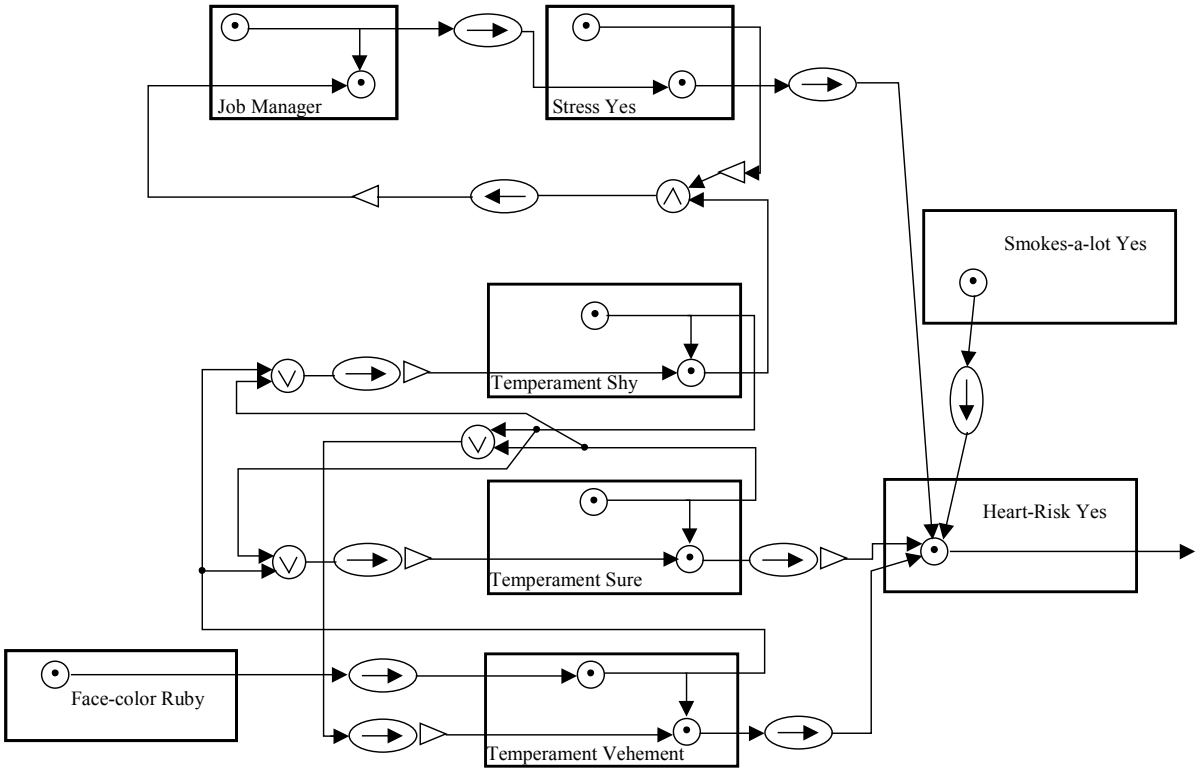

Figure 3: *The network representation of the knowledge base*

4. The hardware implementation

4.1 Building the network

The initial rules are given in a text file, as shown in Figure 2. The file is first parsed by a program, and the corresponding network (see Figure 3) is generated. The network (a directed non-cyclic graph) is built according to the construction rules presented in Section 3.

Then, based on this graph, a VHDL source file is generated. This network can be easily implemented in hardware, because the belief interval values are between 1 and 9 (i.e., on four bits) and all the operators are based on Boolean concepts. This network had, of course, to be adapted to the hardware, in the way explained below.

There are two types of nodes in the network: *operators* and *propositions*. The propositions are virtual nodes - in fact, they contain a group of several operator nodes.

In the hardware implementation, each proposition has two four-bit data registers associated: Support and Plausibility, which are initialised with the values given in the text file containing the knowledge base. Similarly, each DET operator (and only that operator) has two four-bit data registers (that can only be initialised at the beginning of the process: their internal values can not be changed); these registers hold the STRENGTH of the rule. The other operators implement only their own behaviour, according to their definition.

Each operator's behaviour is implemented by combinatorial logic (gates, multiplexers, etc.). There are four-bit data busses that link the network with the PC. On these busses, the network can be initialised and the final results will be read.

The internal graph's representation is translated into VHDL source code, each node in the graph being emulated by a VHDL entity with the architecture (implementing their behaviour) given by the operators definitions.

4.2 Propagating belief interval values through the network

Once the network is built, it is fed with the information about a certain patient, i.e. the PC will send, on the data busses, the initial values of Support and Plausibility for the first proposition. These values will unbalance the network, in the sense that all the operators will produce new values for the Support and Plausibility registers of other propositions.

The propagation of this information constitutes the inference mechanism of the expert system.

On each clock cycle, a new value is computed for several nodes in the network (the nodes that are influenced by the new information introduced, that unbalanced the network). In parallel with the propagation of these values, the PC sends the initial values of Support and Plausibility for the second proposition, and so on. These values will interfere in a constructive manner with the values that are already circulating inside the network; this effect is ensured by the nature of the logic operators.

The propagation process will end when no change occurs in the network, at any node. A special purpose flag detects this. Then, the PC will read back the results from each proposition node and will display them.

5. Conclusions

The hardware implementation of the expert system shell shows a great deal of advantages over classical software implementation, which was also implemented. Even if the network generation phase is a little bit longer (because of VHDL source code generation), the propagation phase speed is much higher and allows also an execution in a parallel manner, that was impossible to achieve on a single-processor computing system. The gain of speed becomes evident when working with large networks, which are not more difficult to build because the construction of the network is 100% automatic.

The network was simulated, tested and implemented on a Xilinx XC4005E FPGA device, using the Xilinx Foundation Series development tools. The only inconvenient is the size of the Xilinx implementation, because we used the behavioural VHDL description of the network. A large amount of space could be saved if using the structural description; this will make the subject of future developments of the project.

6. References

[1] A. Bonarini, E. Cappelletti, A. Corrao, “Belief Revision and Uncertainty: a proposal accepting cyclic dependencies”, Dipartimento de Elettronica, Politecnico di Milano, Report n.90-067, 1990.

[2] C. Cenan, “An expert system shell based on belief revision concepts”. ACAM scientific journal, p.35-45, Cluj-Napoca, 1996.

[3] G.D. Kleiter, Bayesian diagnosis in expert systems, Artificial Intelligence, 1992, No. 54, pp.1-32.

[4] Xilinx, Inc., The Programmable Logic Data Book, 1999.