

Striving for Imprecision...

Fuzzy Knowledge Bases for Business Process Modeling

©1999 Earl Cox



Scianta Intelligence

1289 North Fordham Blvd. Suite A312
Chapel Hill, NC 27517

(919) 678-0477
www.scianta.com

**The real problem is not whether machines think
but whether men do.**

B. F. Skinner
Contingencies of Reinforcement (1969), Chapter 9.

**I believe there are 15,747,724,136,275,002,577,605,653,961,181, 555, 468, 044, 717, 914, 527,
116, 709, 366, 231, 425, 076, 185, 631, 031, 296 protons in the universe, and the same number
of electrons.**

Sir Arthur Stanley Eddington (1882-1944)
Turner Lecture, 1938.

There is a persistent rumor going around that the expert system industry is dead. But, in fact, the old style expert systems, sometimes built using flavors of LISP, sometimes built from the ground up by knowledge engineers using Basic, Pacal, or dialects of C, have simply fused with the technologies of neural networks, genetic algorithms and fuzzy logic. These intelligent systems, often referenced under the computational intelligence banner, are becoming common tools in the quest to build powerful, flexible, and adaptive models of the business world's highly nonlinear problems. Emerging from this witch's brew of intelligent technologies is a new and more powerful breed of knowledge-base system. These are Fuzzy Knowledge Bases, combining both the semantics of information with the imprecision found in nearly all real-world problems.

Fuzzy Associative Memories

A majority of the literature surrounding fuzzy knowledge base structure is concerned with the formulation of control engineering applications. These knowledge bases take the form of an R-dimensional hypercube – one dimension for each variable in the problem space. The edges of these dimensions contain the underlying fuzzy sets. Such knowledge bases are commonly called Fuzzy Associative Memories [see Kosko, 1992]. As a (brief) example, consider the problem of balancing and then maintaining the equilibrium, of an inverted pendulum. We have two input variables – *Theta*, the angle of the pendulum in relation to the vertical (θ), and *DeltaTheta*, the angular momentum of the pendulum ($\Delta\theta$) and one outcome variable – *MotorForce*, the motor current (Δm) which imparts its own angular momentum.

Each of the variables is decomposed into a collection of over-lapping fuzzy sets defining the semantics of some portion of the domain. It is common in control engineering applications to measure rates of change, angles, currents and resistance using the same terms. In our case we chose five fuzzy states: ZE (around Zero), SN (a Small Negative), BN (a Big Negative), SP (a Small Positive), and BP (a Big Positive). Following this approach, *DeltaTheta*, the angular momentum variable (see Figure 1), with a Universe of Discourse (or simply, the Range) of -25 to $+25$, is represented by the following fuzzy sets.

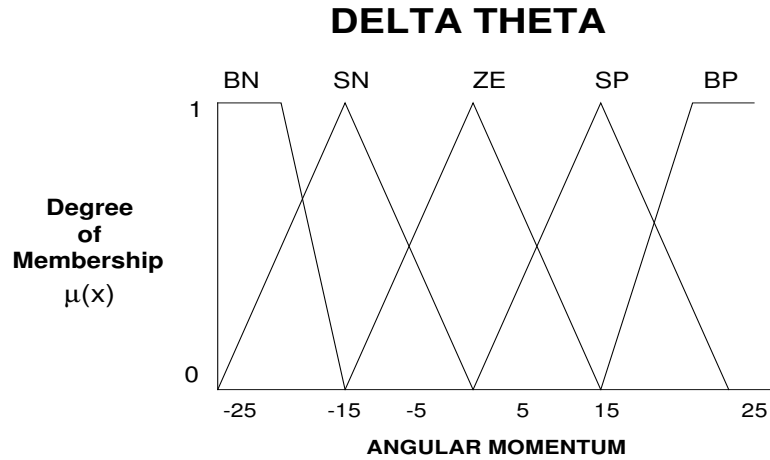


Figure 1. *DeltaTheta*, The Angular Momentum Variable

With the other variables similarly decomposed, we can now write rules for the fuzzy knowledge base, indicating how much motor force should be applied when the pendulum is in a particular state. These rules are expressed as simple if-then statements, such as the following:

If Theta is SN and DeltaTheta is ZE then MotorForce is SP

Which says, “if the pendulum angle is somewhat negative and the angular momentum is around zero, then the motor force should be a small positive.” As Figure 2 illustrates, we can combine all the rules into a two dimensional (5x5) FAM (for clarity only the core set of rules is shown).

		<i>Theta</i> (θ)				
		BN	SN	ZE	SP	BP
<i>Delta Theta</i> ($\Delta\theta$)	BN			BP		
	SN		ZE	SP	ZE	
	ZE	BP	SP	ZE	SN	BN
	SP		ZE	SN	ZE	
	BP			BN		

Figure 2. The Inverted Pendulum 5x5 FAM

Because fuzzy sets overlap (a point in the domain is in the BN as well as the SN sets to different degrees), a single value usually causes several rules to fire. A fuzzy knowledge base handles this condition by effectively running all the rules in parallel. Each rule contributes some evidence for the final state of the system (such as the MotorForce outcome). This contribution based on evidence is done by combining outcome fuzzy sets after they have been scaled to account for the degree of truth in the rule’s antecedent. When all the rules have been executed, we find the outcome value by defuzzifying the final, outcome fuzzy set. Figure 3 Illustrates how a FAM Knowledge Base, coupled with a feed back loop (to measure the pendulum’s state after the motor force has been adjusted) is used to control the inverted pendulum.

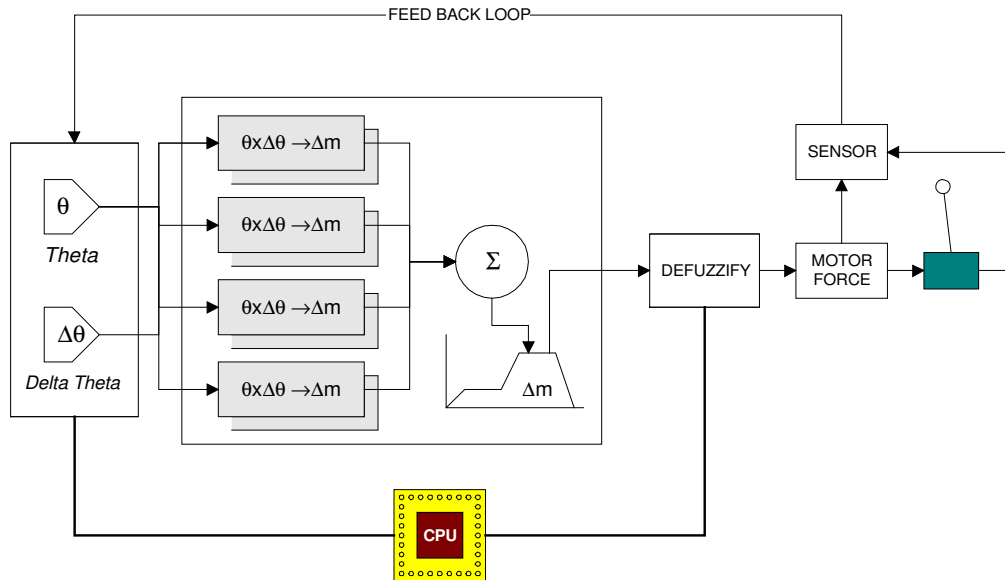


Figure 3. Running a FAM Knowledge Base

All this is well and good – if our problems in decision support, expert system development, and business process modeling were as simple and straight forward as control engineering. Unfortunately when we turn to the world of modeling complex, highly non-linear business problems, the rather shallow architecture of the Fuzzy Associative Memory used in control applications fails to provide the robustness we need. In building real-world fuzzy expert systems, our tools must work on knowledge bases that have a much deeper architecture, incorporate many of the techniques from traditional Artificial Intelligence, and must connect to a wide variety of external data sources (such as databases, Hyper Text Markup Language (HTML) and Extensible Markup Language (XML) pages, spread sheets, and flat files).

Fuzzy Knowledge Bases for Decision Support

The difference between the FAM matrix used in control applications and a Fuzzy Knowledge Base used in expert and decision support systems is schematically illustrated in Figure 4. A business model is generally decomposed into one or more small knowledge bases, often called Policies. Policies contain variables (with their fuzzy sets) and the collection of if-then-else rules.

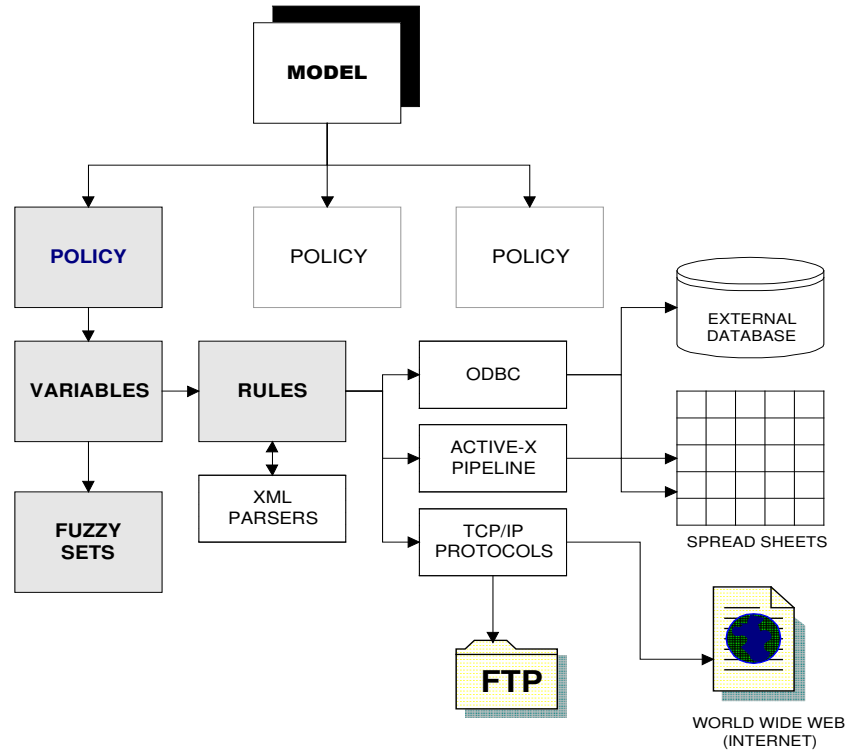


Figure 4. A Modern Fuzzy Knowledge base

Rules in the fuzzy base must handle much more than the numeric input values in control engineering applications. And, as the diagram in Figure 4 shows, the source of information used by a fuzzy knowledge base includes massively large corporate and public databases, spread sheets and flat files, as well as the rich and highly unstructured contents of the Internet. To this end, new knowledge base systems are increasingly able to find as well as store knowledge using XML (and its industry specific dialects, such as CML, the Chemical Markup Language and the evolving BSML, the Biosequence Markup language used in gene mapping and the Human Genome Project.)

Why Fuzzy Knowledge Bases are Different

Of course, this kind of tight integration with external data sources is nothing new for knowledge engineers working with conventional expert systems. It is the breadth, power, and robustness of the knowledge base structure that separates a fuzzy system from its conventional cousins. In addition to the if-then rule forms that we have come to expect in non-fuzzy (or crisp) knowledge bases a fuzzy knowledge base contains a rich array of new forms.

Form	What is Does	Example
<i>Unconditional Rules</i>	establish constraints on the outcome variable's decision space. They also provide a default value for the outcome if none of the conditional rules fire. Unlike the assignment statement, its counterpart in a conventional expert system, unconditional fuzzy rules	Our price must be High Our price must be Low Our price must be around $2 * MfgCosts$

actually create a solution space in the outcome variable. As you can see, unconditional rules provide a direct method of encoding often contradictory and conflicting knowledge.

Hedged Semantics

provide new ways to use a basic collection of static and dynamic fuzzy sets. Using hedges – such as *very*, *somewhat*, *near*, *around*, *positively*, and *not* - a knowledge engineer can express a rule with functional operators that intensify, dilute, and approximate existing fuzzy sets. Hedges change the meaning of fuzzy sets in the same way that adjectives and adverbs change the meaning of nouns and verbs.

If the competition.price is **not very** High then Our price should be **near** the competition.price

Relational Operators

allow the comparison of model states based on approximate metrics. Fuzzy rules use this technique to extend the concepts of greater than, less than and equal found in non-fuzzy systems. The fuzzy relation much greater, much less, and so forth, can also use hedges to intensify or dilute the comparison strength. In a fuzzy rule, the degree to which the comparison is true is used to assess how much evidence exists in the consequent proposition.

If costs are **much greater** than expenses then profits are near zero.

If costs are **very much greater** than expenses then profits are near zero.

Lead-Lag and subscripted variables

are intrinsic parts of any business-based fuzzy modeling system. We want to write rules that compare the state of the system across time. Fuzzy rules also handle fuzzy time relationships so that we can write statements like

```
if avg(delta(inventory,t-10,t))  
is small
```

which measures the average change in the inventory level across the previous ten time periods and checks if it is *small* to some degree.

If backorder(t-1) + QOH(t) is less than inventory(t) then shortfall is increased;

Dynamic Fuzzy sets (fuzzy numbers)

are bell, trapezoidal, or triangular shaped fuzzy sets representing approximate quantities or fuzzy numbers. Any fuzzy reasoning system attempting to deal realistically and robustly with the real world must be capable of creating new, dynamic fuzzy sets.

Our price must be **near 2*MfgCosts**
If costs are **near avg(expenses)**
then markup is **about .5*InterestRate**

Where Fuzzy Knowledge Bases are (or Should be) the Same

On the other hand, a modern, business-oriented Fuzzy Knowledge Base incorporates several features found in more conventional knowledge base systems. Two of these are fairly important. Chief among the principles common to both fuzzy and conventional knowledge base systems is the concept of knowledge acquisition through backward chaining. This kind of fundamental machine reasoning is an essential component of any expert and decision support system. The second concept shared by crisp and fuzzy knowledge bases is the use of hybrid or complex if-then-else rules. We briefly consider both of these.

Backward Chaining

Backward chaining, also called goal-directed reasoning, is used to find the value of a variable in the model. As an example, in a fuzzy policy that finds a value for product line margins, we might have the rule,

```
If Costs are High and Volume is Low
then Margins are Reduced
```

We need a value for Costs in order to execute the rule. The variable Costs is used as an intermediate goal (the primary goal is Margins). All the rules that have Costs in their outcome or consequent expressions are collected and executed.

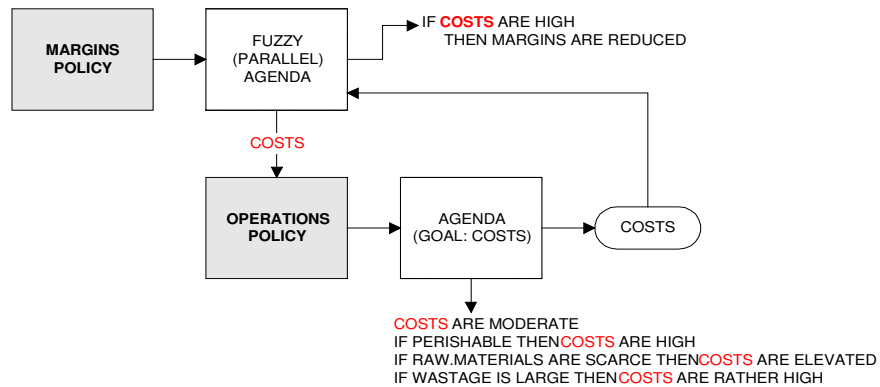


Figure 5. Fuzzy Backward Chaining for Costs

Since all the fuzzy rules in a policy are executed in parallel, backward chaining must, of course, use rules stored in another policy. In the example illustrated by Figure 5, all the rules in the Operations Policy are collected and run as a fuzzy model. If any of these rules have variables without values – such as Perishable or Wastage – each variable is selected, in turn, as a goal, and another recursive backward chaining process is initiated.

Mixed (Hybrid) Rules

Fuzzy Associative Memories are pure fuzzy logic machines – all the rules are fuzzy rules. The same is almost always true for the current breed of fuzzy logic development tools on the market. They were



developed by engineers or control scientists with little thought to the nature of business policy analysis. Modern Fuzzy Knowledge Bases incorporate both pure fuzzy rules as well as “hybrid” rules. As an example, the following is an actual rule from a project risk assessment system (with the outcome actions slightly modified).

```
For each project where duration is long and budget is high;
If project.costs are near project.budget then do;
    If project.RevisionCount is not small then
        Apply Rule R1;
        Execute KillProject(project.projID);
        Execute FireProjectmanager(project.ProjManager);
    Else
        If (project.budget/project.duration) is low then
            Perform Project_Statistics(project.*);
        End if;
    End if;
End if;
```

Mixed, hybrid rules provide the backbone for fuzzy knowledge bases used to model business processes, government policies, manufacturing operations, and a host of other applications where the system must handle simultaneous strategies, execute a variety of consequent actions, represent complex conditional decisions, read and write data to a variety of local and remote media, as well as handle textual, categorical, and ordinal in addition to continuous data.

A Handful of Fuzzy Expert Systems

Compared to the neural network and genetic algorithm marketplace, there are relatively few fuzzy knowledge base development tools available today that were not originally designed and developed for control engineering purposes. Here is a small sampling of some tools that are used to create and deploy fuzzy knowledge bases for expert and decision support systems.

Fuzzy CLIPS is a version of the CLIPS expert system development shell originally designed by NASA and now available through the NASA/Johnson Space Center. Fuzzy CLIPS integrates fuzzy logic and approximate reasoning concepts with more conventional uncertainty measurements at the rule level. Fuzzy CLIPS is available free of charge. You can download this product from the Internet at: <http://ai.iit.nrc.ca/fuzzy/>. I should point out, that the idea of fuzzy logic as an engineering tool is so strongly ingrained even among knowledge engineers, that the on-line demonstration for Fuzzy Clips is a fuzzy shower controller.

FLOPS (Fuzzy Logic Production System) was developed in the early 1980's at the Kemp-Carraway Heart Institute by Douglas Tucker and Bill Siler. It was originally used for medical image and pattern analysis. FLOPS is a powerful, general purpose fuzzy reasoning system. FLOPS is the only development tool that can work not only with numeric fuzzy sets but also with non-numeric fuzzy quantities. Dr. William Siler, who evolved and maintains the FLOPS system, is arguably one of today's most experienced and influential practitioners of real-world fuzzy modeling. He can be reached at wsiler@aol.com.

On the West Coast, in the heart of the Silicon Valley, Zaptron Systems, Inc. an off-shoot of Apronix, an early control engineering company, has recently re-emerged as a vendor of fuzzy data mining and expert system tools. While the flavor of their software still bears the imprint of their control

Striving for Imprecision – Fuzzy Knowledge Bases



engineering heritage, Zaptron's product line appears headed towards a rich suite of general purpose fuzzy modeling tools. They can be reached on the Internet at <http://www.zaptron.com/index.shtml>.

Metus Systems, my own company, provides a comprehensive suite of fuzzy and conventional knowledge base development tools. These tools combine strategic backboard services with hybrid fuzzy reasoning, backward and forward chaining. Metus Tools is built on an extensive set of design tools for building complex, business-related process modeling systems in today's Rapid Application Development (RAD) environments such as Visual Basic, Dephi, PowerBuilder and Visual C++. Metus Systems can be found on the Internet at <http://www.metus.com>.

References

Kosko, Bart, **Neural Networks and Fuzzy Systems**, Prentice-Hall, Inc, 1992

For more information or to schedule a presentation call (919) 678-0477 or visit www.scianta.com



©2004 Scianta Intelligence, LLC
AR-PA-003