Rule Chaining in Fuzzy Expert Systems

Lawrence O. Hall
Dept. of Computer Science and Engineering, ENB 118
University of South Florida
Tampa, Fl. 33620
hall@csee.usf.edu

Abstract

A fuzzy expert system must do rule chaining differently than a non-fuzzy expert system. In particular, any rule that can fire with a particular linguistic variable in its consequent must fire before any rule whose antecedent conditions depend upon the resultant fuzzy set value of the consequent linguistic variable is allowed to fire. The dependent rules would be considered in a chain with the fuzzy rules which generate or assert the needed fuzzy linguistic variable. A recent paper by Pan, DeSouza, and Kak points out that a version of the FuzzyCLIPS expert system shell does not operate with chained fuzzy rules as one would expect. They introduce FuzzyShell which is described as the only known shell to have the expected fuzzy rule chaining performance. In this paper, we show several approaches to obtaining the desired behavior in FuzzyCLIPS. Further, a potential pitfall with the FuzzyShell approach to dealing with chaining is pointed out.

Keywords: fuzzy expert systems, chaining, decision making

1 Introduction

There are a number of available approaches to fuzzy expert system shells [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. According to [14] none of the early fuzzy shells addressed the idea of rule chaining in the expected way. This lead Pan, DeSouza, and Kak [14] to introduce FuzzyShell which does address rule chaining as one might expect in a fuzzy expert system.

Rule chaining occurs when one or more rules have an antecedent condition which matches an assertion in the consequent of one or more rules. In classical expert systems, where facts are true or false, rule chaining is a simple proposition. If one of the rules which has the required antecedent condition in its consequent fires, the rules which depend on it (are in a chain with it) can then fire. Figure 1 shows an example of three crisp rules in a chain.

Fuzzy expert system rule firings will result in additions to working memory of fuzzy linguistic variables and associated fuzzy sets. It is possible that more than one rule with

the same fuzzy linguistic variable in the consequent will fire. Consider Figure 2. Rules 1 and 2 may both fire if the temperature is 57ºF and this has a non-zero membership in both antecedent fuzzy sets as would be the case in this example. The combination of the firing of rules 1 and 2 will result in a fuzzy set for morning that is distinct from those obtained by firing either rule by itself. Therefore, Rule 3 should only fire after Rules 1 and 2 have been applied so that the resultant fuzzy set associated with morning takes into account the effects of both Rule 1 and 2. However, utilizing classic expert system rule firing strategies (depending upon the conflict resolution strategy) it is likely that Rule 3 will always fire before both rules 1 and 2 have fired. This is not what someone using a fuzzy expert system would expect.

In particular, FuzzyCLIPS, which is a freely available, portable and reasonably robust extension of the CLIPS expert system shell [1, 15], does not guarantee the desired behavior given the three rules in Figure 2. FuzzyShell, on the other hand, will allow rules 1 and 2 to fire first and create the appropriate composite fuzzy set before it is utilized in firing Rule 3.

There are other fuzzy shells, such as in the fuzzy logic toolbox of Matlab [9], that are not set up to allow chaining of fuzzy rules at all. They are primarily designed for 1-level fuzzy control rules. Fuzzy rule chaining has been used in applications, for example in a fuzzy controller for maintaining distance between cars [16].

In this paper, we will show three ways to enable FuzzyCLIPS to also have the desired or expected behavior in the face of fuzzy rule chaining. It is also pointed out that the approach used in FuzzyShell can have unexpected results in the case of fact retraction. The rest of the paper is organized as follows. Section 2 discusses the problem of choosing which rule to fire

```
Rule 1: IF day temp < 50F THEN day requires sweater
Rule 2: IF day requires sweater THEN day is bus-day
Rule 3: IF day is bus-day THEN departure-time is 7:45a.m.
```

Figure 1: A 3 rule chain with Rule 2 depending on Rule 1 and Rule 3 depending on Rule 2.

```
Rule 1: IF morning temperature is below55  THEN morning is cold
Rule 2: IF morning temperature is about55-75  THEN morning is moderate
Rule 3: IF morning is moderate THEN bike time is fast
```

Figure 2: A 2 rule chain with Rule 3 depending on Rule 1 and Rule 2.

for FuzzyCLIPS and FuzzyShell in more detail, Section 3 describes two simple approaches to allow FuzzyCLIPS to have the desired behavior and a more complex approach, as well as a low-level change to the program. Section 4 discusses a hidden issue with the choice made in FuzzyShell and Section 5 is a summary.

2 Rule firing approaches

Production system type expert systems work on a match, recognize and act cycle as shown in Figure 3. Both FuzzyCLIPS and FuzzyShell are production systems. In the match stage all rule antecedents are matched against working memory. In the recognize stage all rules that can fire are considered to be in conflict and a *conflict resolution* strategy is employed to rank or order them.

Commonly used conflict resolution strategies are lexicographic analysis (LEX) and means ends analysis (MEA) [17, 18]. Both strategies place a premium on the time recency of facts which match a rule's antecedent when determining what rule to fire next. They give preference to rules whose antecedent (partially) matches the most recently asserted facts. In Figure 4 we show 4 rules in the FuzzyCLIPS format. The fuzzy sets they use are shown in Figure 5. The example is nearly identical to that used in [14] with only minor modifications in fuzzy set boundaries causing the differences.

In FuzzyCLIPS when the rule starter is fired and asserts the facts about the robot's mission and position both rule1 and rule2 will be able to fire. Next rule1 will fire and then rule3 will be able to fire. Rule3 will be first on the agenda of rules to be fired because its antecedent matches the fact most recently asserted (or added) to working memory.

```
       Knowledge
         Base                                  Inference
                                                Engine
      ┌──────────┐                        ┌──────────────────┐
      │          │                        │ 1. Match: facts  │
      │  Rules   │────────────┐           │ against rules.   │
      │          │            ╲           │                  │
      └──────────┘             ╲          │ 2. Recognize:    │
                                ╲         │ rules that can   │
                                 ╲        │ fire, rank rules.│
       Working                    ╲       │                  │
       Memory                      ──▶    │ 3. Act: fire top │
                                          │ ranked rule.     │
      ┌──────────┐            ──▶         └──────────────────┘
      │          │           ╱                     │
      │ Facts.   │          ╱                      │
      │ The Current         ◀──────────────────────┘
      │ Situation.│
      └──────────┘
```
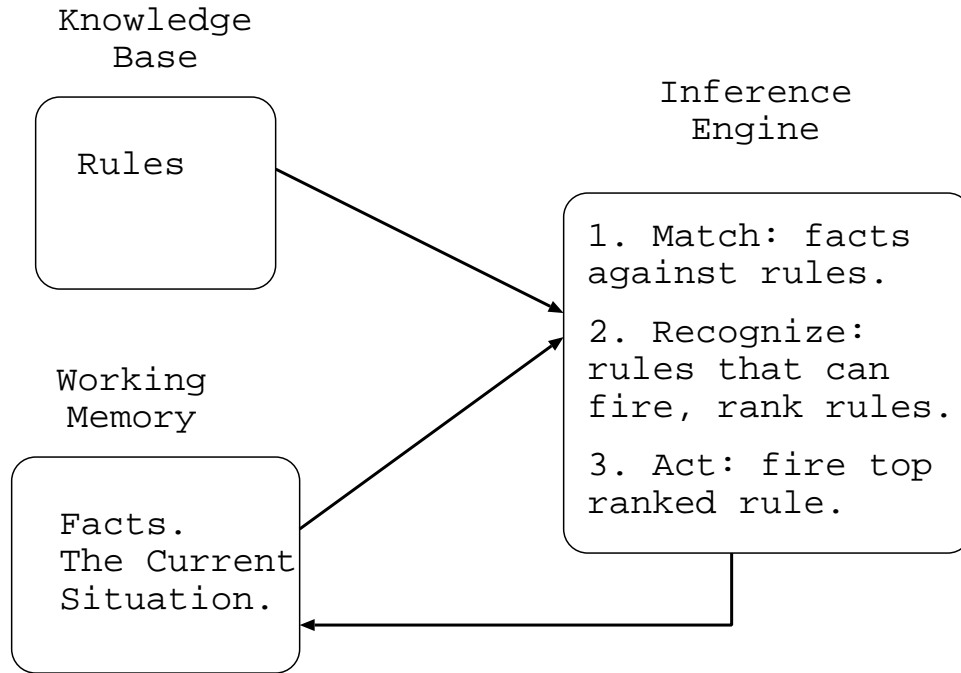
Figure 3: A production system. Rules fire until no more rules match the facts in working memory. Only 1 rule fires per cycle.

However, with fuzzy rules one would expect rule2 to fire before rule3 since it has the effect of modifying a fuzzy set used in the antecedent of rule3. In FuzzyShell rule1 and rule2 will fire before rule3. This was accomplished by a modification to the underlying pattern matching structure or the Rete network [19]. The Rete network is a fast pattern matching facility which is commonly used to speed up rule firing in production systems [17, 19, 18].

To be fair, it must be noted that when using FuzzyClips version 6.04a [1] after rule3 fires rule2 will fire. Since rule2 modifies a fact that is used by rule3, rule3 will fire again. The end result is a set of facts in working memory that look slightly different than if rule1 and rule2 had fired and then rule3 fired, but are, in fact, equivalent in this simple example.

Figure 6 shows a partial set of rules with 3 levels of chaining for a tennis player who is attempting to determine when to go into work based on what time the scheduled tennis match will occur. The fuzzy sets are shown in Figure 7. The time of the match (in Florida at least) is affected by temperature and wind. On cold, windy days one plays earlier to

4

```
(defrule rule1
 (robot (mission ?x) (sensor 2D-vision))
 (2D-vision (pixel-position-x left) (pixel-position-y near))
=>
 (assert (goal (mission ?x) (orientation NWW))))

(defrule rule2
 (robot (mission ?x) (sensor 2D-vision))
 (2D-vision (pixel-position-x med-left) (pixel-position-y far))
=>
 (assert (goal (mission ?x) (orientation NW))))

(defrule rule3
 (goal  (mission obstacle-avoidance) (orientation NW))
=>
 (assert (turn-angle right-20))
 (assert (distance-to-travel medium)))

(defrule starter
=>
(assert (robot (mission obstacle-avoidance) (sensor 2D-vision)))
(assert (2D-vision (pixel-position-x med-left) (pixel-position-y far))))
```

Figure 4: A subset of fuzzy rules for robot control based on those in the FuzzyShell paper by Pan, et. al.

minimize discomfort. However, with the set of rules in Figure 6 different work start times will be chosen based on whether all fuzzy rules fire at a given rule level of this chain. In fact the order the rules are given to FuzzyCLIPS will cause a different start time to be determined. If you imagine a person using this set of rules to decide when to expect a workers arrival, it can vary under the exact same external conditions simply because someone did rule base maintenance and re-ordered the rules.

Consider the case in which the temperature is 50°F and the wind speed is 9.5 mph. Now under the default FuzzyCLIPS conflict resolution procedure, rules temp1, temp2, wind1 and wind2 can all fire. However, with the rules above in FuzzyCLIPS temp2 and wind2 would fire and rule courttime1 would fire. Next, if the value for the last time played is over-week,

rule worktime1 will fire. However, there is more information about the shape of the fuzzy set tennis-time which has not been utilized because rules at the higher level have not yet fired. One would expect rules temp1, temp2, wind1 and wind2 to fire first and then rules courtime1 and courtime2 to fire and finally the worktime rules to provide the expected start-time. Hence, the defuzzified time to start work will be different than most people conversant with fuzzy expert systems would expect[1].

However, if all rules that act on a given fuzzy set are fired before any rule dependent on the resultant fuzzy set (e.g. a rule further *down* in the rule chain) fires, the final start time will be the same for the identical (or even highly similar) external conditions. The results will also mirror those that a user who understands the technology would expect.

3 Making FuzzyCLIPS rule chains work as expected

There are three different possibilities that can be used to enable FuzzyCLIPS to fire all fuzzy rules with a shared linguistic variable in their consequents before allowing a rule to fire that uses the same linguistic variable in its antecedent.

3.1 Utilizing salience

The first approach to ensuring that all rules which have the same fuzzy consequent and can fire, do in fact fire, before other rules which utilize the associated fuzzy linguistic variable makes use of the FuzzyCLIPS saliency feature. Each rule in a FuzzyCLIPS rule base can be assigned a salience value which allows the knowledge base builder to indicate rules that should fire before other rules. A rule salience can be an integer in the range $[-10000, 10000]$. By default a rule salience is 0. Rules with higher salience values go to the top of the agenda and will be fired before rules with lower salience.

---

[1]FuzzyClips version 6.04B at http://ai.iit.nrc.ca/IR_public/fuzzy will provide the expected final time. However, it will fire rules at a level multiple times (using the default conflict resolution) where one firing is what you might expect. The multiple rule firings are extra work and non-intuitive in the best case.

To force the desired rule behaviour, a convention of giving fuzzy rules that assert the same fuzzy linguistic variable the same salience value can be used. The salience value of the rules sharing a linguistic variable in their consequent simply must be higher than that of any fuzzy rule which is in a chain at the proceeding level. So, any rule which utilizes the fuzzy linguistic variable from the consequent of a set of fuzzy rules will have a lower salience and will not fire until all rules which can affect the fuzzy set value have fired.

In our first example from the last section we simply add:

(declare (salience 10))

as the first condition in rule1 and rule2 from Figure 4 which will cause them to fire before rule3. Now the results are as one would expect, matching the performance of FuzzyShell.

For the rules in Figure 6 the desired behaviour can also be obtained with the use of salience. Rules temp1, temp2, wind1, and wind2 would have

(declare (salience 10))

added as their first antecedent condition. Rules courttime1 and courttime2 would have

(declare (salience 5))

added as their first antecedent condition. Then the rules will fire in the expected order with complete values for fuzzy sets being determined before the rules are used in the next level of chaining. This slight modification to the rules will cause the results to match those of FuzzyShell.

This solution puts some burden on the knowledge base designer or knowledge engineer. One might argue, "what about long chains", this requires a number of distinct saliencies. While this is true, it can also be argued that long fuzzy chains are unlikely. They are unlikely because each conclusion will get progressively fuzzier (in most cases) fairly quickly resulting in conclusions that are so vague or fuzzy they are of little use.

One can reasonably expect a fuzzy chain to be no more than 10 levels with a range of 2 to 5 levels being the most likely. If the chain is more than 10 levels, the conclusion is going to be very vague as it will get fuzzier at each level. This means that the fuzzy rules

require at most 10 distinct salience values and more likely between 2 and 5. The same salience values can be used for every fuzzy chain in a knowledge base. So, that means that at most 10 distinct salience values are necessary for fuzzy chains regardless of the size of a knowledge base.

This approach does potentially restrict how salience is used in non-fuzzy rules. However, the reservation of 2-5 salience values seems a reasonable restriction.

3.2 Changing conflict resolution strategy

Another solution is to utilize the breadth strategy that currently is one of the 7 existing conflict resolution strategies [18]. The breadth strategy orders rules so that those matching the most recent facts go to the bottom of the agenda. Hence, rules in a chain with a fuzzy linguistic variable will not fire until all rules that act on the variable and can fire have fired. This is because newly firable rules are added to the bottom of the current rule agenda.

If the breadth strategy for conflict resolution is not appealing for use throughout the set of rule firings, the set-strategy command [18] can be used in the right-hand side of rules to change strategies. However, this approach has drawbacks in that it is neither seamless nor does it allow anything except the breadth strategy to be used for ordering the firing of the the first non-fuzzy rule or fuzzy rule not in a chain.

While the breadth strategy clearly has the drawback of requiring the knowledge engineer to integrate it into the knowledge base, it may be acceptable in some applications. Certainly, anytime the breadth strategy is effective for knowledge base this simple change can be used to get the desired performance from fuzzy rule firing. This strategy is likely most effective with less than 500 rules and many fuzzy rule-sets have been described as effective with less than 500 rules [20, 21, 22, 23, 24].

3.3 Modules in FuzzyCLIPS

Another possible way of getting all fuzzy rules which act on a particular fuzzy fact to fire together is to group such sets of fuzzy rules into modules [18]. FuzzyCLIPS allows rules to be defined in encapsulated modules. It would be necessary for the knowledge base designer to encode sets of fuzzy rules into appropriate modules. It will also be necessary for them to be able to set up the focus stack to switch among modules properly and it will be necessary to correctly import and export deftemplates. If there are many fuzzy chains in a very large knowledge base, this approach may be unwieldy. However, many fuzzy rulebases are relatively small (under 500 rules) [20, 23].

Control switches out of the current module when there are no rules left on the agenda for that module and there is a module on the focus stack. Each module gets its own agenda. When a set of fuzzy rules is done firing the next module with fireable rules must become the focus of attention which means the order of module firing must be known to set up the focus stack. A set of rules that make use of modules can be found at `http://www.csee.usf.edu/~hall/module.clp` together with a firing trace. The rule base is equivalent to that shown in Figure 4.

3.4 Modifying FuzzyCLIPS

An alternative approach that does not require changing the underlying Rete network of FuzzyCLIPS is to define a new conflict resolution strategy for FuzzyCLIPS. This approach requires modifications to the code of FuzzyCLIPS, but only for rule ordering on the agenda. The new strategy, call it the *fuzzy* strategy, will be slightly different from the 7 current mutually exclusive strategies. It will be allowed to operate simultaneously with any one of the 7 strategies (though its use with breadth is redundant). It gets applied before any current conflict resolution strategy orders the fireable rules. The *fuzzy* strategy simply does a temporary change to the saliencies of any rules on the agenda which have a fuzzy linguistic

9

variable in their consequent, which was in the set of facts added to working memory by the last rule firing. It makes the saliency of the affected rules the highest which will cause them to move to the top of the agenda. The rule saliencies are not permanently changed and will revert to their original values for use the next time the rules are fireable.

This approach allows any of the 7 current conflict resolution strategies to be used in the absence of fuzzy chaining. It will seamlessly cause the desired behaviour of all rules which have a linguistic fuzzy variable in their consequent to fire before a rule that utilizes the linguistic variable in its antecedent fires. Further, it requires no change to the underlying pattern matching structure and is simply a choice that the knowledge base designer can utilize, if desired.

There are a few issues with ordering rules on the agenda that need to be acknowledged here. First, it is certainly possible for someone to produce a FuzzyCLIPS knowledge base of, for example, more than 5,000 rules in which every rule is on the agenda in every cycle. This would make ordering fuzzy rules on the agenda quite time-consuming, however, just ordering the agenda is going to be essentially the same amount of time. In general most rule-based systems will have less than 50 rules on an agenda with between 1 and 10 the most likely number given the way knowledge engineers typically construct knowledge bases. Many fuzzy rule based systems, especially, will not have a large number of rules.

Perhaps more serious is the issue that FuzzyCLIPS, which is built upon CLIPS, is much more general than FuzzyShell and it allows functions to be in the right hand side of rules. These functions may assert a fuzzy fact based on quite complex formulae. It may not be possible a priori to determine whether a fuzzy fact will be asserted when it is embedded in a function on the right hand side of rule. Hence, the proposed solution does not apply to functions on the right hand side of fuzzy rules. It requires that fuzzy consequent be added to working memory **only** by the assert command. Such a limitation with fuzzy rules seems rather minor.

Figure 8 shows an example that can be problematic in FuzzyCLIPS with the agenda ordering scheme[2]. It also poses problems for other fuzzy expert systems. Consider $A2$ is asserted which partially matches both $A1$ and $A3$ and consider that Y is asserted. Now, R1 and R3 will go on the agenda. If R3 fires first our agenda ordering procedure will cause the desired behavior. Otherwise, it will not and the combination of B1 and B3 will not apply to R4. The order of rule firing in FuzzyCLIPS is indeterminate. Hence, for this example to chain correctly that use of salience with R3 is necessary.

Adding run-time overhead to the ordering of rules on the agenda is a concern. During the process of loading the rules, a table can be created with the rule number of each fuzzy rule and for each fuzzy rule, a list of other fuzzy rules which share its fuzzy consequent. When a fuzzy rule becomes highest on the agenda, the table of fuzzy rules which share the consequent of the top-ranked fuzzy rule can be used to appropriately bump the saliency of the other fuzzy rules with its consequent which are on the agenda. This requires only one simple pass through the rules on the agenda which will be a minimal time cost in the usual case of only a small number of rules being fireable in any given match, recognize, act cycle. This strategy assumes that the fuzzy consequents are not embedded in functions.

4 Pitfalls of FuzzyShell

A potential problem for the knowledge base designer who is using FuzzyShell can occur when fuzzy facts are retracted from working memory. Consider the two rules (using FuzzyCLIPS-like syntax) shown in Figure 9. Rulea retracts the fuzzy fact x is A which is also used by Ruleb.

Assume the facts x is A and y is B are in working memory and that the fuzzy set for y is $very$ B overlaps with y is B. Now both Rulea and Ruleb can fire. As defined in FuzzyShell when Rulea fires before Ruleb, Ruleb will be fired in the background to get the final fuzzy

---

[2]Thanks to the anonymous referee for pointing out a similar example.

set for the fuzzy linguistic variable r. However, one of the facts required for Ruleb to fire is retracted by Rulea. Anyone not intimately familiar with the workings of FuzzyShell would not expect Ruleb to fire in this case. Indeed it does not seem to be reasonable behavior for a rule to fire after a fact required by it is retracted. Ruleb will not fire under the approaches to dealing with chains of fuzzy rules proposed for use in FuzzyCLIPS.

Of course any time a fact is retracted there are issues of non-monotonic reasoning [25] to be addressed. Should facts that were added to working memory due to the presence of a retracted fact be retracted themselves? Sometimes the answer is yes. However, in other uses the answer is no and CLIPS rule base designers commonly utilize fact retraction. Our point is that the above feature of FuzzyShell is only a pitfall under some viewpoints.

5 Summary

In this paper we have discussed chaining in fuzzy rule based systems. In fuzzy systems it is desirable for all rules which affect the value of a linguistic variable to fire before that variable is used in the antecedent of another rule. Rules are said to be in a *fuzzy* chain with other rules (higher in the chain) which have fuzzy linguistic variables in their consequents which match a rule's antecedent fuzzy variables. FuzzyShell [14] makes some complex modifications to the underlying Rete pattern matching network to enable fuzzy chaining to work as expected.

The authors of FuzzyShell note that no other fuzzy expert system shell correctly handles chaining in fuzzy rule-based systems. We show three approaches to correctly handling chaining in FuzzyCLIPS. The approaches can be used with FuzzyCLIPS 6.04a and require no modifications to the underlying pattern matching algorithm. Further, a mechanism that modifies the rule agenda ordering algorithm while allowing the current 7 different conflict resolution approaches of FuzzyCLIPS to be used is described. This approach also solves the fuzzy chaining problem in FuzzyCLIPS in a manner transparent to the user.

Finally when facts are retracted, we pointed out a potential pitfall of the choices made in FuzzyShell to handle fuzzy chaining. This potential pitfall does not exist in any of the FuzzyCLIPS solutions to the problem that are given here.

References

[1] R.A. Orchard. Fuzzy clips version 6.04 users guide. Technical report, National Research Council, Canada, http://ai.iit.nrc.ca/fuzzy/fuzzy.html, 1995. Institute for information technology, Ottawa, Ontario, CA K1A OR6.

[2] L.O. Hall and A. Kandel. Fess: A re-usable fuzzy expert system. In A. Kandel, editor, *Expert Systems in the Fuzzy Age.* CRC Press, Boca Raton, Fl., 1991.

[3] K. P. Adlassing and G. Kolarz. Representation and semiautomatic acquisition of medical knowledge in cadiag-1 and cadiag-2. *Comput. Biomed. Res.*, 19:63–79, 1988.

[4] J. Buckley and D. Tucker. Second generation fuzzy expert system. *Fuzzy Sets and Systems*, 31:271–284, 1989.

[5] P.L.K. Jones. Reveal: An expert systems support environment. In R. Forsythe, editor, *Expert Systems: Principles and Case Studies.* Chapman and Hall, 1984.

[6] K.S. Leung, W.S. Wong, and W. Lam. Applications of a novel fuzzy expert system shell. *Expert Systems: Int. J. of Knowledge Eng.*, 6(1):2–10, 1989.

[7] Z.A. Sosnowski. Flisp – a language for processing fuzzy data. *Fuzzy Sets and Systems*, 37:23–32, 1990.

[8] I.B. Turksen, Y. Tian, and M. Berg. A fuzzy expert system for a service centre of spare parts. *Int. J. Expert Systems Applications*, 5:447–464, 1992.

[9] The Mathworks, Natick, MA., www.mathworks.com. *Fuzzy Logic Toolbox Users Guide*, version 2 edition, 1999.

[10] M. Schneider, A. Kandel, G. Langholz, and G. Chew. *Fuzzy Expert Systems Tools*. Wiley, Wiley, England, 1997.

[11] G. Chew, M. Schneider, A. Kandel, and G. Langholz. Incorporating membership functions and certainty factors in fuzzy expert systems. *Journal of Computer-Aided Civil and Infrastructure Engineering*, pages 213–222, 1994.

[12] L. Godo, R. Lopez de Mantaras, C. Sierra, and A. Verdaguer. Milord: the architecture and the management of linguistically expressed uncertainty. *International Journal of Intelligent Systems*, 4(4):471–501, 1989.

[13] E. Vombrack, M. Togai, Y. Toki, and A. Miyata. A fuzzy development tool for easy prototyping: Til shell 3.0 be. In *Proceedings of 1995 IEEE International Conference on Fuzzy Systems*, pages 41–42, 1995.

[14] J. Pan, G.N. DeSouza, and A.C. Kak. Fuzzyshell: A large-scale expert system shell using fuzzy logic for uncertainty reasoning. *IEEE Transactions on Fuzzy Systems*, 6(4):563–581, 1998.

[15] J. Giarratano and G. Riley. *Expert Systems: Principles and Programming*. Boston: PWS Publishing, third edition, 1998.

[16] R. Holve, P. Protzel, J. Bernasch, and K. Naab. Adaptive fuzzy control for driver assistance in car-following. In *EUFIT '95*, pages 1149–1153, Aachen, Germany, 1995.

[17] L. Browstone, R. Farrell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5.* Addison-Wesley, Reading, MA, 1985.

[18] Gary Riley. *CLIPS Reference Manual, Volume I: Basic Programming Guide*, 1998. Version 6.10, http://www.ghg.net/clips/download/documentation/.

[19] C.L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[20] M. Sugeno and T. Yasukawa. A fuzzy-logic based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31, 1993.

[21] C-F. Juang, J-Y. Lin, and C-T. Lin. Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 30(2):290–302, 2000.

[22] Yan-Qing Zhang, M.D. Fraser, R.A. Gagliano, and A. Kandel. Granular neural networks for numerical-linguistic data fusion and knowledge discovery. *IEEE Transactions on Neural Networks*, 11(3):658–666, 2000.

[23] L.I. Kuncheva. How good are fuzzy if-then classifiers? *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 30(4):501–509, 2000.

[24] L.O. Hall and P. Lande. Generation of fuzzy rules from decision trees. *Journal of Advanced Computational Intelligence*, 2(4):128–133, 1998.

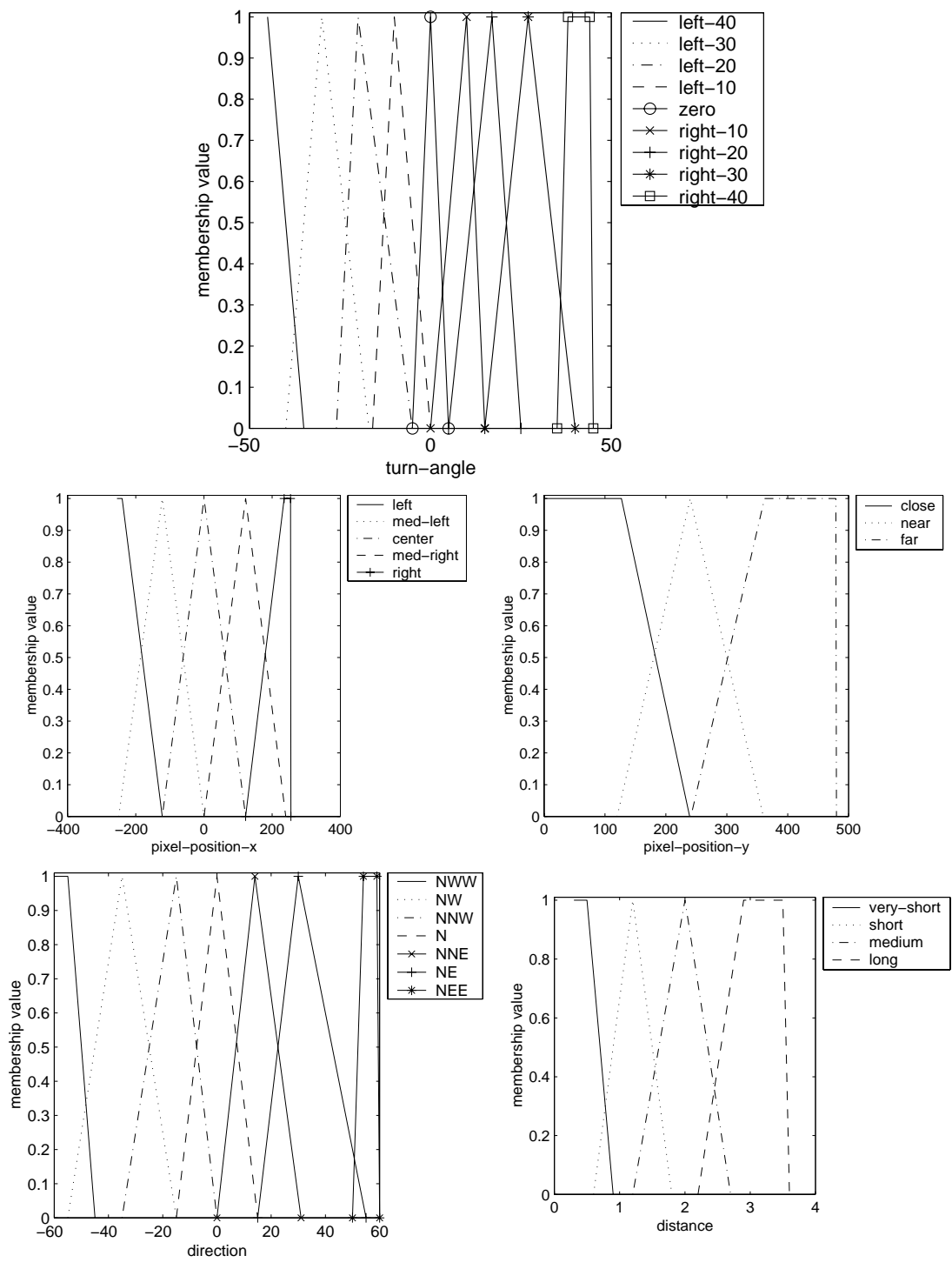[25] M. Davis. The mathematics of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:73–80, 1980.

Figure 5: Fuzzy sets of fuzzy rule example.

```
(defrule temp1                        (defrule temp2
 (day (temperature  below-50))          (day (temperature above-50))
=>                                       (day (temperature below-70))
 (assert (day (tennis-temp cold)))     =>
                                        (assert (day (tennis-temp moderate))))


(defrule wind1
   (day (windspeed above-10))
=>
   (assert (day (tennis-wind windy)))


(defrule wind2
   (day (windspeed above-5))
   (not (day (windspeed above-10)))
=>
   (assert (day (tennis-wind moderate))))


(defrule courttime1
   (day (tennis-temp cold))
   (day (tennis-wind windy))
=>
  (assert (day (tennis-time early))))


(defrule courtime2
   (day (tennis-temp cold))
   (day (tennis-wind moderate))
=>
   (assert (day (tennis-time usual-late))))


(defrule worktime1
   (day (tennis-time early))
   (play (last-time over-week))
=>
   (work (start-time early)))


(defrule worktime2
   (day (tennis-time usual-late))
=>
   (work (start-time usual)))
```

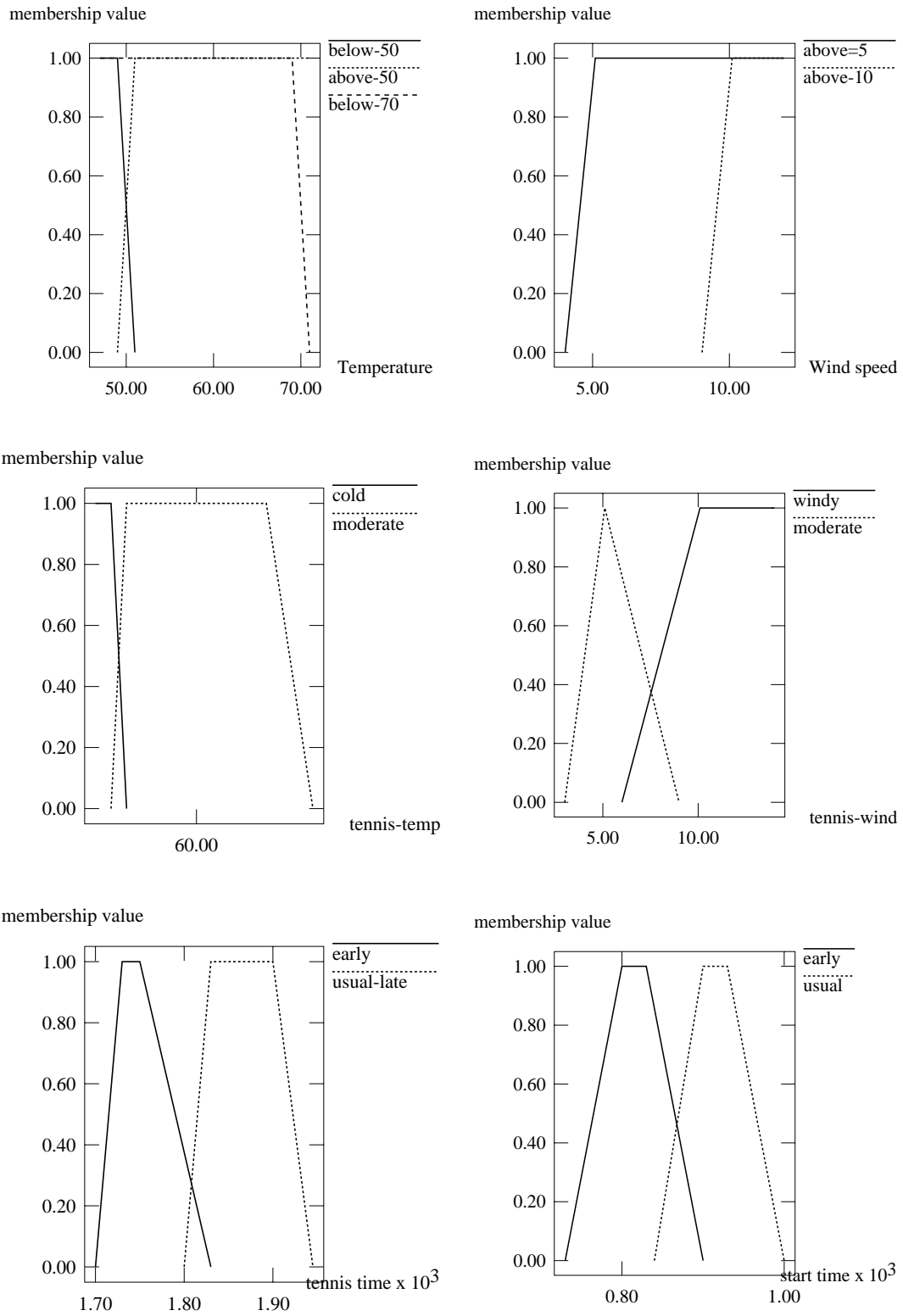Figure 6: A set of fuzzy rules with three levels of chaining.

Figure 7: Fuzzy sets for tennis example fuzzy rules. Time is on 24hr. scale.

```
R1: If A1 Then B1
R2: If A3 and X Then B3
R3: If Y Then X
R4: If B2 Then Z1
```

Figure 8: a set of special case fuzzy rules in which A and B are fuzzy variables with fuzzy sets $\{A1, A2, A3\}$ and $\{B1, B2, B3\}$ are fuzzy sets on the appropriate fuzzy variables and the middle fuzzy set partially overlaps its neighbors. X and Y are crisp facts and Z1 is a fuzzy fact.

```
(defrule Rulea                        (defrule Ruleb
  f1 <-  (x is A)                         (x is A)
    (y is B)                              (y is very B)
=>                                    =>
 (retract ?f1)                          (assert r is very C))
 (assert (r is C))
```

Figure 9: Two fuzzy rules which have non-intuitive results in FuzzyShell.