Chapter 4

Tutor Modeling

*Everything should be made as simple as possible, but not simpler.*

Albert Einstein.

This chapter describes the essentials of the proposed tutor model. After a brief introduction that puts the pedagogical concepts and requirements in proper perspective, the related work on tutoring module is described. A brief overview of fuzzy finite state machines is presented. Described next is the rationale for using fuzzy state machine for tutor modeling. The proposed fuzzy automaton is presented along with its ANFIS based implementation and testing. Finally, the chapter concludes with the summary and discussion.

4.1 Introduction

The tutoring component provides a model of the teaching process. The tutoring module, commonly referred as the pedagogical module has to take several decisions like when to review, when to present a new topic, and which topic to present. It uses information from the student model to determine what aspects of the domain knowledge should be presented to the learner. This information may be new material, a review of previous topics, or feedback on the current topic. One pedagogical concern for an ITS is the selection of a meta-strategy for teaching the domain. Meta-strategy is concerned with the higher level issues of topic selection and sequencing. Once the meta-strategy is selected, low level issues, such as the exact example to use, must be decided. In this work we have delegated these low level functions to the expert module also referred as adaptive courseware.
generator. However, for the sake of completeness and to put all the pieces in
context, both low level and high level issues are described below.

4.1.1 Low level issues

Conventionally, a tutor must decide the content of the material to be
presented to the student. This involves decisions about the topic, the lesson, and
the feedback.

*Topic selection:* To select a topic to present, the tutor must examine the student
model to determine the topics on which the student needs to focus. Many
possibilities exist for the most appropriate topic on which a student should work.
For example, if the meta-strategy indicates that review is in order, the tutor will
select a topic the student has already "learned." On the other hand, if new
information is to be presented, the tutor will choose a topic that the student does
not yet know.

*Lesson generation:* Once the topic has been selected, a lesson must be generated
for presentation. The grain size of the lesson is determined by the domain and the
domain author. Whatever the granularity of the lesson generated, it is important
that the difficulty be appropriate for the student's level of ability, which can be
determined from the student model.

*Feedback:* Most tutors work smoothly as long as students get everything right.
Problems arise when the student has difficulties and needs help from the tutor. In
these situations, the tutor must determine the kind of feedback to provide. The
issue of how much help to provide to the student is also a very complex issue as
too little feedback can lead to frustration and floundering [Anderson, 1993] while
too much feedback can interfere with learning [Kasahara et al., 1994]. Once the
system decides how much feedback to give, it must determine the content of the
advice. The feedback should contain enough information so that the student can proceed to the next step in understanding the lesson. Furthermore, the advice given to the learner should be appropriate for his ability level.

4.1.2 Meta-strategy selection

High level strategy selection in ITSs has not received the same amount of attention as the low level decisions. This is not to say that meta-strategies have not been researched. To the contrary, educational research has identified many potential teaching strategies for use by an ITS [Kearsley, 2003]. Examples of these kinds of strategies include spiral teaching [Bruner, 1992] and the Socratic method. However, implementing these meta-strategies in an ITS has proven a formidable problem. Most ITSs do not explicitly identify the strategies they are using for teaching and implicitly implement one of the well-known strategies [Major & Reichgelt, 1992]. A better method is to use the student model to select an appropriate strategy from those maintained by the system. Ideally a student's model could track the instructional strategies that are most effective for teaching him. However, because most systems do not have multiple teaching strategies, student models have not been designed to provide selection information. Thus, representing multiple strategies explicitly and the control knowledge to select among them is beyond the capabilities of most current systems.

Another obstacle in representing multiple teaching strategies is the limitations imposed by other components of the ITS in addition to those placed by the student model. In particular, the difficulty of representing knowledge impedes the ability to explicitly represent teaching strategies. For example, the Socratic method requires substantial "common sense" knowledge beyond what is maintained in a domain knowledge base.
Empirical studies have demonstrated that the most effective way for students to learn is to work alone, face-to-face with a qualified tutor, equipped with instructional material and laboratory equipment [Bloom, 1984]. The main reason why human tutors are so effective is because they are able to tailor feedback (respond to student's questions about the subject matter, determine when students need help and identify the type of help needed) to students. Unfortunately, the need for qualified tutors makes this preferred form of instruction very costly and rare. ITSs aim to provide a learning experience for each student that approaches the standard of learning that he would receive in one-to-one tutoring from an expert tutor equipped with all the necessary training aids. The ideal ITS would be one modelled on a human tutor. However, trying to identify how human tutors individualize feedback to students is a complex topic because the process depends on many different factors that the tutor takes into account on any one occasion. For example, the tutor has several different ways of responding, and chooses an appropriate one depending on:

i. what model of the learning process the tutor has in mind for each outcome of the instruction,

ii. what steps the tutor thinks the student needs to follow to reach the outcome,

iii. where the tutor thinks the student is in the sequence of steps the student needs to follow, and

iv. the tutor's diagnosis of why the student might have made a mistake (which is based on the model of the student's abilities, background knowledge, and personality, that the tutor has in mind).

All this varies with different kinds of subject matter, how easy or difficult the tutor thinks the subject matter is, and how much time and effort the tutor
thinks the subject matter deserves. It also varies with the tutor's estimate of the motivational level of the student.

The main goal of ITSs is to provide a learning experience for each student that approaches the standard of learning that he would receive in one-to-one tutoring from an expert tutor. One method of achieving this is to provide individualization of instructions (providing instructions that match the student's level of ability). Individualization of instructions is important because students may have different prior knowledge, learning experience, particular interests, motivation, or personality, and these factors affect how a human tutor interacts with the student.

4.2 Conventional Teaching-Learning Process & the Related issues

Figure 4.1 shows an overview of the conventional instruction process. A teaching strategy is selected based on the knowledge type being taught and the learner's current depth of knowledge. The actual instruction then changes the learner's level of knowledge depth by either adding new knowledge or "deepening" existing knowledge. Testing is conducted to evaluate the success of the instruction and the potential need for a repetition or modification of instruction.
to inform the student that a particular knowledge type has not been mastered. At this point, the student alone is responsible for changing the situation and obtaining mastery, without the help of an instructor. A better instruction method would be to personalize the instruction based on the background and the progress of each individual student via either human or computer based tutors. A personalized method of instruction is characterized by

1) Learner-centered pacing,
2) The ability to retake tests until mastery is demonstrated,
3) Immediate feedback,
4) Small units of instructional material,
5) The use of peer proctors to administer feedback and testing, and
6) Optional lectures [Calhoun, 1975].

For computer-based training systems, the computer can take over the role of peer proctors in the form of online-testing and grading, and lectures can be placed online as well. The other principles can be directly applied to the development of ITS by providing learners with different versions, different difficulty levels, etc., based on their performance and previous knowledge.

In the following sections, we review the two approaches that have emerged in the area of software training research [Olfman & Mandviwalla, 1994] that try to improve end-user training:

1) Adaptation of the training material content and
2) Adaptation of the training material presentation.
4.2.1 Adaptation based on Content

Adaptation of the content implies that different learners receive different versions of the same lecture. Common approaches are 1) adapting the level of difficulty appropriate to level of understanding, 2) adapting the level of depth according to stated learning goal of the learner, which can range from high level overview to in depth discussions, 3) adding appropriate references to existing knowledge [Merrill et al., 1996]. Content adaptation has also the benefit of allowing the ITS to support a variety of learning styles such as textual vs. visual or audio presentation forms of the same training material, or even different versions of the content to support learning styles such as concrete vs. abstract. [Davidson, et al., 1992].

4.2.2 Adaptation Based on Sequence

Besides adapting the content of the training material, another well-researched area is the adaptation of the instruction sequence. Bernstein [1998] found that two approaches to lesson sequencing seem to be work well on the Web. These approaches are based on either behaviorism or constructivism theory [Brandt, 1997]. The former guides a student through predefined steps (system control), and the latter provides all the resources and lets students construct knowledge themselves (learner control). Niemiec [Niemiec et al., 1996] conducted a meta-analysis of the research on learner-control. They found that most studies used a combination of learner control features, had a system-controlled control group, and administered an immediate posttest. At first glance, the overall average result of near zero effect of learner vs. system-control might surprise researchers and discourage any further investigation in this area. However, the way research on learner-control has been conducted has been criticized [Reeves, 1993], and any
meta-analysis in this area in fact compares “apples and oranges” and thus has to be
examined very closely. The following paragraphs present an overview of research
in this area. Schnackenberg et al. [1998] presented a thorough overview of the
research on learner- vs. system-control. Arguments in favor of learner-control are
that

- learners know their own instructional needs best,
- learner-control can help students become independent learners, and
- learners construct their own knowledge in the context of their own needs
  and experiences and require control over the learning process to do so.

Critics claim that the learner-control distracts learners because it forces
them to interrupt their learning and pay attention to the sequencing of material
[Schnackenberg, et al., 1998]. Others [Murray, 1998] claim that beginners are
unable to make the right sequence choices, because they lack the background
knowledge necessary to make educated decisions. Similarly, Lieberman and Linn
[Lieberman & Linn, 1991] found that novices should benefit from system control,
but more advanced students could benefit from learner-control. Others go even
further claiming that the degree of learner-control should depend on the learner’s
familiarity with the topic as well as the learner’s motivation, aptitude, and attitude
[Merrill, et al., 1992].

In order to avoid the constant questions about how to proceed in learner-
control systems, an experiment by Schnackenberg [Schnackenberg et al., 1998]
tried asking the learners in advance about the amount of instruction and practice
they desired. Matching and mismatching the learners to their preferred amounts,
the study showed that although subjects preferred the lean version of the
instructional material, scores were higher for subjects who took the full version, which shows that learners are not making the best choices when asked directly.

Most learner-control studies ignore the proficiency level of the learners. For example, McGrath [McGrath, 1992] examined the question of whether learners do benefit from learner-controlled systems. She examined the impact of hypertext, CAI and program-control and found differences in score, navigation, and time spent. On the other hand, Rieman and others [Rieman et al., 1996] showed in their experiments that students did not follow a linear viewing pattern when they had full control; instead, no specific sequence was followed in what they call exploratory learning. Supporting both these views, Allinson [Allinson, 1992] reported in her study that some subjects used a more linear navigation approach, and others preferred self-determined hypertext navigation. The effect of navigation on performance was examined by Melara [Melara, 1996] who found no performance differences when students used a hierarchical organized system compared to a network structure. However, others like Tennyson [Tennyson, 1981] and later Goforth [Goforth, 1994] found performance differences in that learner control is more effective than system-control, but Young [Young, 1996] supported this finding only for learners with high self regulated learning strategies, not for others, showing a potential link of navigation habits and level of expertise.

Based on this discussion, the research on the effectiveness of learner- vs. system-control seems to be inconclusive at best. It might be the case that the two approaches, learner-control and system-control, work well for different students, but at this point, the research regarding the optimal balance of learner- vs. system-control is inconclusive. It would appear that the best system would adapt itself to
the learner by providing both the behaviorist and the constructivist approach based on the learner’s preference and personality type [Tan, 1996].

4.3 Related Work to Tutor Modeling

This research is focused on building a better tutor for any domain by replacing traditional model-tracing feedback in an ITS with a performance-based feedback mechanism. The proposed tutor model is novel because it is based on the observation of an experienced human tutor and captures tutorial strategies independent of the domain. In this context, a tutorial content is the equivalent of breaking down problems into simpler steps and then asking new questions before proceeding to the next step. Studies indicate that experienced human tutors provide the most effective form of instruction known [Bloom, 1984]. They raise the mean performance about two standard deviations compared to students taught in classrooms. Intelligent tutoring systems can offer excellent instruction, but not as good as human tutors. The best ones raise performance about one standard deviation above classroom instruction [Anderson et al., 1995]. Although Ohlsson [Ohlsson, 1993] observed that teaching strategies and tactics should be one of the guiding principles in the development of ITSs, incorporating such principles in ITSs has remained largely unexplored [McArthur, et al., 1990]. Hence, ITSs where a model of both the student and the tutor are created in an effort to improve performance was the natural extension to model-tracing tutors. Several researchers have developed ITSs that model the tutor as well as the student, such as Graesser et al. [Graesser et al., 2001] with AutoTutor, Heffernan [Heffernan, 2001] with Ms. Lindquist and Rosé et al. [Rosé et al., 2001] with Atlas-Andes. In studying what makes a tutoring session successful, Murray [Murray, 1998] identified principles for effective teaching. One important principle was that tutors should not offer
strong hints or apply rules to problems themselves when students make mistakes. Students miss the opportunity to learn how to solve a problem when they are given an answer and are not allowed to reason for themselves. [Merrill et al., 1992] compared the effectiveness of human tutors and intelligent tutoring systems. It is concluded that a major reason that human tutors are more effective is that they let the students do most of the work in overcoming impasses, while at the same time provided as much assistance as necessary. [Merrill et al., 1992] argue that the main thing human tutors do is to keep students on track and prevent them from following "garden paths" of reasoning that are unproductive and unlikely to lead to learning. Modeling, coaching, and scaffolding are described by [Collins et al., 1991] as the heart of cognitive apprenticeship, which the authors claim "help students acquire an integrated set of skills through processes of observation and guided practice." An important part of scaffolding is fading, which entails progressively removing the support of scaffolding as the student demonstrates proficiency [Collins et al., 1991].

4.4 Fuzzy Finite State Machines

4.4.1 Introduction

A finite automaton also called a finite state machine (FSM) or sequential machine is a dynamic system operating in discrete time that transforms sequences of input states at the input of the system to sequences of output states produced at the output of the system. The sequences may be finite or countably infinite. The transformation is accomplished by the concept of a dynamically changing internal state. At each discrete time, the response of the system is determined on the basis of the received stimulus and the internal state of the system. At the same time, a new internal state is determined, which replaces its predecessor. The new internal

Chapter 4: Tutor Modeling
state is stored in the system to be used the next time. An automaton is called a finite fuzzy state automaton (FFSA) when its states are characterized by fuzzy sets, and the production of response and next states is facilitated by appropriate fuzzy relations [Klir & Yuan, 1995].

The main difference of FFSA with respect to traditional FSM is that transitions in the automata are not triggered by crisp events but by fuzzy variables, and state transitions are fuzzy as well. It immediately results that, at any time, the whole system is not necessarily in one and only one well-defined state, but it may well be in more states at the same time, each one associated with its own membership value. State transitions are therefore smoother and slower. As a consequence, decision process based on states is simplified, each one designed for a different target specific of that state. In practice, FFSA achieve approximately the same speed of traditional FSA with simplicity of design (especially for complex systems). So far, the theory of FFSA has been tested on some applications in the field of control, for instance to design the Motion and Leg Coordination Controls of a walking hexapod [Berardi, et al., 1997], where FFSA have been used as a way to specify the cooperation between individual legs.

4.4.2 Operation of Fuzzy State Automata

This section describes theory and operation of FFSA, and in particular how FFSA can be converted to fuzzy descriptions [Berardi, et al., 1997]. The theory will be described by means of the example shown in Figure 4.2. This is a simple example with only one input variable, but the diagram could well be part of a more complex FFSA.
Furthermore, the theory applies also to FFSA with more than one input. An FFSA looks very similar to a traditional FSA, in the sense that it can be represented as a collection of fuzzy states $S_j$ (namely, the circles), connected by fuzzy transitions (namely, the arrows). Each fuzzy transition is labeled by a fuzzy expression which can be as complex as desired ($A, B, C, D$ are traditional fuzzy sets). As in a traditional FSA, a state represents univocally the operating conditions of an FFSA, at any given instant. But, unlike FSA, a system need not be in only one state at a time. Each state $S_j$ is therefore associated with a fuzzy state activity $\mu_{S_j} \in [0,1]$, which represents "how much" the system is in that particular state. The state activity is somehow equivalent to the degree of membership of a fuzzy variable, yet note that the state in an FFSA is not a fuzzy variable. If we associated a state activity to states in a traditional FSA, it would be:

$$\mu_{S_j} = 1; \text{ for the active state}$$

$$= 0; \text{ for all other states}$$

with the constraint that the total activity:

$$\sum_j \mu_{S_j} = 1,$$
meaning that the system is always in one and only one well defined state S_j (the
one with \( \mu_{S_j} = 1 \)) and in no other state. Constraint second applies also to FFSA,
therefore activity can be distributed among several states and can partially move
from one state to another (possibly more than one), but the total activity shall
always be constant and equal to 1.

FFSA can be of different types: time-independent and time-dependent; the
former can either be synchronous or asynchronous, which somewhat reflect
synchronous and asynchronous FSA, respectively. They differ only in the way
state activity moves from one state to another, according to the degrees of
membership of the fuzzy transitions. Roughly we could say that:
i. in asynchronous, time-independent FFSA, state transitions may take place as
soon as inputs vary (yet transitions are not so abrupt as in traditional FSA);
ii. in synchronous, time-independent FFSA, state transitions are computed in a way
similar to asynchronous FFSA, but they are applied only at the next clock cycle;
iii. in time-dependent FFSA, there is always an intrinsic delay (usually larger than
the clock period, if any) between input variations and the corresponding state
transitions.

In time-independent FFSA, the activity moves from each state to one or
more other states, as a function of the present state activity and the degrees of
membership of the state transitions, but independently of time (hence the name).
State activity may “move" only along state transitions. In more details, a time-
independent FFSA (either synchronous or asynchronous) can be translated into a
more traditional fuzzy description made of one rule per each transition, where each
rule is in the form:

IF (STATE is oldstate) AND (fuzzyexpression) THEN (STATE is newstate)
where the term (STATE is Sj) has, by definition, a degree of membership equal to the state activity $\mu_{Sj}$, whereas the AND, OR and NOT implicators have the traditional meaning of any fuzzy system [Wang, 1992]. Two rules with the same consequent are supposed to be connected by an OR implicator. Transitions starting from and ending into the same state must also be taken into account. For instance, for the example shown in Figure 4.2, the equivalent time-independent fuzzy description is made of five rules:

**IF** (STATE is SA) **AND** (x is NOT A) **THEN** (STATE is SA)

**IF** (STATE is SA) **AND** (x is A) **THEN** (STATE is SB)

**IF** (STATE is SB) **AND** (x is B) **THEN** (STATE is SB)

**IF** (STATE is SB) **AND** (x is C) **THEN** (STATE is SC)

**IF** (STATE is SB) **AND** (x is D) **THEN** (STATE is SD)

A constraint has to be placed on the membership functions $\mu_i(x)$ associated with all the transitions exiting from a state, to let the FFSA operate properly [Klir and Yuan, 1995]:

$$\sum_{i \in T_j} \mu_i(x) \approx 1 \quad \forall x \in U_x, \forall j$$

where $T_j$ is the set of transitions exiting from state $Sj$ and $U_x$ is the universe of discourse of the input variable $x$. In traditional FSA, the transitions exiting from a state must be both exhaustive (meaning that the logic sum of all transition labels must always be true) and mutually exclusive (meaning that no two transitions must be active at the same time). Similarly, in FFSA, the above constraint is equivalent to exhaustivity, but there is no equivalent of mutual exclusion.
4.5 Fuzzy FSM for Tutor Modeling

In this section we describe tutoring as a kind of information cycling system. The task of the intelligent tutoring system (ITS) can be briefly characterized as a problem of instructional material sequencing, diagnosing and responding to (specific) user requests. In the ideal case, the material is constructed on demand, and intelligent re-mediation can be performed at any level. The material sequencing and diagnosing has two aspects. A logical mistake should be corrected by providing the appropriate information. The strategy of presentation may depend on the user. The extent to which the content and the presentation strategy should be controlled by the user profile is one of the crucial question of ITS. In our system, each lesson is presented by cycling elements of information, in the form of definitions, statements, concepts, examples, explanation and quizzes. A final test allows the user to leave the lesson and to pass to the next one. A user, and his path through the system, is characterized by his performance. The performance of the learner is decided as poor, fair, good or excellent based on tests of memory, concept understanding and misconceptions. These parameters control the number of lesson cycles and the lesson exit criterion. It means that each user is associated with a recommended number of cycles for each lesson. The other meaningful parameter is the kind of domain, the learners' level and the amount of information presented at a time. Our goal is to build tutoring systems that can be effective in the introductory courses at undergraduate level. We are further interested in using interactive educational tools to introduce key vocabulary and basic concepts in the subjects from engineering and technology disciplines.

From our classroom experience, we have seen that students can benefit substantially from interactive information cycling. The targeted audience was the
lower level undergraduates. When students are exposed to vocabulary training before an instructor-led lesson, difficult concepts can be introduced more easily. This idea has been applied in classrooms in which the majority of the students are required to take mathematics or computer science but do not need to understand deeper principles such as proofs. This system is useful for students who need to master an introductory understanding of the basic vocabulary and concepts. Even within the same classroom, there are differences in presentation needs. The differences among users in such a classroom setup is the skill level rather than the background level. In accommodating a variety of presentation needs, the user oriented system reflects skill level in both reducing redundancy for more skilled users and maintaining appropriate level for those less skilled.

Figure 4.3. A Fuzzy Finite State Machine (Lesson Level)
It is proposed to model the tutoring process as a fuzzy finite state machine (FFSM) as shown in the Figure 4.3. The rationale for selecting a fuzzy modeling approach is humanistic nature of the problem. It is universally accepted that a fuzzy logic based system can better model a humanistic system [Zadeh, 1973]. The reason for finite state machine based modeling is due to the cyclic nature of teaching-learning process. Further, the cognitive state of the learner also changes from learning experience. State 1 is the start state, wherein a lesson is aggregated according to the students’ previous knowledge level. It is decided to have the granularity of the material at the lesson level. Thus, the FFSM will, in general, undergo several transitions till it reaches the final state. Note that the input to each state is fuzzy. This means the system at a particular instance will be in all the four states with different degrees of membership. The resulting state is obtained by aggregation of all the states.

Consider that, due to initialization, the system is in state 1 initially. A lesson is generated and presented to the learner followed by the test. The results of this test is fuzzified in terms of the performance of the student in that particular lesson through student model. The performance as a fuzzy singleton may be poor, fair, good or excellent. If the performance comes out to be poor, the system remains in the same state. However, now another lesson is generated and presented to the learner along with the feedback. One important point to be considered is: what if the students’ performance does not reach ‘excellent’ level, ever after several iterations? In such cases, maximum number of iterations should be completed with different material and the student should be allowed to switch over to the next lesson. However, his such performance should be noted, and used to
guide the selection of next lesson. This is in consonance with the behavior of a human teacher.

At the unit level and the topic level, the tutor simply transits from one state to another state. This scenario is depicted in Figure 4.4.

![Tutor Finite State Machine (Unit level & Topic level)](image)

**Figure 4.4. Tutor Finite State Machine (Unit level & Topic level).**

Although domain knowledge representation is detailed in a separate chapter in this thesis, a brief description of that model is in order. As shown in the Figure 4.4, a subject domain is divided into several units. Each unit is divided into several topics. Each topic is further divided into several lessons. Lessons are composed dynamically and presented to the learner. It is at this level that the information cycling takes place. After a particular lesson has been mastered, another lesson is generated and presented. This is depicted in the figure by lesson blocks. When all the lessons for a particular topic are mastered, the system moves to the next topic.
This is depicted by topic ovals in the figure. The sequencing of the topic is already fixed as we have a tutor-controlled system. Similarly, when all the topics for a unit are mastered, the system moves to the next unit. Again, the sequencing of the units is fixed. Likewise all the units are completed.

The advantages of this approach are:

i. The tutoring is domain independent. It can be applied to any subject domain that can be organized by the domain expert in the form of reusable, sharable learning assets.

ii. Being tutor-controlled, it prevents the learners' wandering into the hyperspace of the domain.

iii. It guarantees a minimum level of proficiency achievement for the learner.

iv. It is adaptive to the learners' performance level.

v. It is similar to human-like tutoring.

vi. It prevents cognitive overloading to the learner.

However, following drawbacks are equally applicable:

i. Not very learner-friendly due to tutor-controlled nature.

ii. Fixed cycling of lessons, topics and units may not be suitable to certain domains.

4.6 Implementation and Testing

As described in the previous sections, a tutor can be modeled by a fuzzy finite state machine. For implementation we employ adaptive network-based fuzzy inference system (ANFIS). The rules are:

IF (STATE is 1) AND (PERFORMANCE IS poor) THEN (STATE is 1)

IF (STATE is 1) AND (PERFORMANCE IS fair) THEN (STATE is 2)
A Fuzzy-Neural Intelligent Tutoring System For The World Wide Web

IF (STATE is 1) AND (PERFORMANCE IS good) THEN (STATE is 3)
IF (STATE is 1) AND (PERFORMANCE IS excellent) THEN (STATE is 4)

And so on. The complete set of nine rules is shown in Table 4.1.

Table 4.1. Rules for Fuzzy FSM based Tutor model.

<table>
<thead>
<tr>
<th>Current State</th>
<th>Performance</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>'one'</td>
<td>Poor</td>
<td>'one'</td>
</tr>
<tr>
<td>'one'</td>
<td>Fair</td>
<td>'two'</td>
</tr>
<tr>
<td>'one'</td>
<td>Good</td>
<td>'three'</td>
</tr>
<tr>
<td>'one'</td>
<td>Excellent</td>
<td>'four'</td>
</tr>
<tr>
<td>'two'</td>
<td>Fair</td>
<td>'two'</td>
</tr>
<tr>
<td>'two'</td>
<td>Good</td>
<td>'three'</td>
</tr>
<tr>
<td>'two'</td>
<td>Excellent</td>
<td>'four'</td>
</tr>
<tr>
<td>'three'</td>
<td>Good</td>
<td>'three'</td>
</tr>
<tr>
<td>'three'</td>
<td>Excellent</td>
<td>'four'</td>
</tr>
</tbody>
</table>

Figure 4.5. ANFIS as a Fuzzy Finite State Machine.
Here a memory element through feedback is added to the basic ANFIS. The state output is fed as one of the inputs to the ANFIS as shown in the Figure 4.5. The MATLAB Fuzzy Logic Toolbox has been for testing the proposed tutor model [MathWorks, 2000]. The toolbox provides a GUI based tool to define input-output variables, membership function selection and the rule base design. A Sugeno model with two inputs and one constant output is constructed. Input for transition from one state to the other comes from the student module and they reflect the current performance of the student as poor, fair, good and excellent. Nine rules, as shown in Table 4.1, are formulated and a Sugeno-type fuzzy inference system is built. The ANFIS utility provided under Matlab has certain limitations [MathWorks, 2000]. One of them is that the ANFIS generates individual output membership function for each rule i.e. membership function sharing is not allowed in the output. However, in our case the output has to have only four fuzzy sets to be used in the subsequent course generation module. To work around this problem we utilized the input-output data pairs to generate five different fuzzy inference systems with different membership functions. The training method adapted was hybrid of back propagation with least square error criteria. 100 epochs were used for training. After training the 12-rule ANFIS, it was tested with 45 input-output data pairs. The system with minimum testing error was selected and parameters from this model were used in the fuzzy system designed using the 9-rule rulebase.

The details of the inference system are given below:

<table>
<thead>
<tr>
<th>ANFIS Parameters and Methods for Tutor FFSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Type:</td>
</tr>
<tr>
<td>Sugeno with two inputs one output.</td>
</tr>
<tr>
<td>Number of input fuzzy sets:</td>
</tr>
<tr>
<td>Three.</td>
</tr>
<tr>
<td>Number of output fuzzy sets:</td>
</tr>
<tr>
<td>Four.</td>
</tr>
</tbody>
</table>

Chapter 4: Tutor Modeling
**AND method:** Product.  
**OR method:** Maximum.  
**Defuzzification method:** Weighted average.  

<table>
<thead>
<tr>
<th>Membership function type:</th>
<th>Generalized bell.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter Values</strong></td>
<td><strong>a</strong></td>
</tr>
<tr>
<td>For 'one'</td>
<td>0.2484</td>
</tr>
<tr>
<td>For 'two'</td>
<td>0.1330</td>
</tr>
<tr>
<td>For 'three'</td>
<td>0.2583</td>
</tr>
</tbody>
</table>

**Input 'Performance' membership function details:**  
**Membership function type:** Generalized bell.  
**Parameter Values**  
| For 'Poor'                | 0.0962 | 2.003 | 0.1512 |
| For 'Fair'                | 0.0937 | 2.005 | 0.3944 |
| For 'Good'                | 0.1330 | 2.000 | 0.6307 |
| For 'Excellent'           | 0.1862 | 2.000 | 0.8976 |

**Output 'NewState' membership function details:**  
**Membership function type:** Constant.  
| For 'one'                 | 0.1604 |
| For 'two'                 | 0.3331 |
| For 'three'               | 0.6191 |
| For 'four'                | 0.8939 |

This system is tested using the rule viewer. Screenshots 4.1 to 4.6 depicts ANFIS model, ANFIS structure, the membership functions, the fuzzy rule verification and the training data plot respectively. As depicted in Screenshot 4.6, an average testing error of 0.049438 was achieved against 0.0884 with arbitrary selection of membership functions.
A Fuzzy-Neural Intelligent Tutoring System For The World Wide Web

Screenshot 4.1. ANFIS for FFSM Tutor Model.

Screenshot 4.2. ANFIS Structure for FFSM Tutor Model

Chapter 4: Tutor Modeling
A Fuzzy-Neural Intelligent Tutoring System For The World Wide Web

Screenshot 4.3. Generalised Bell Membership functions for ‘Performance’

Screenshot 4.4. Generalised Bell Membership functions for ‘State’

Chapter 4: Tutor Modeling
A Fuzzy-Neural Intelligent Tutoring System For The World Wide Web

Screenshot 4.5. FFSM Rule Verification showing the rule

\[
\text{if state is 'one' & performance is 'good' then}
\]
\[
\text{newState is 'three'.}
\]

Screenshot 4.6. ANFIS-based FFSM training and testing plot.

Chapter 4: Tutor Modeling
4.7 Summary & Discussion

Curriculum sequencing and selection of appropriate teaching strategy are the important tasks a tutor module should carry out at the higher level while the topic selection, lesson generation and providing feedback are the main activities that need to be handled as low level strategies. After taking brief review of the literature related to tutor modeling, the operation of fuzzy finite state machine is presented. The idea behind describing the fuzzy finite state machine operation is to bring out two important features namely the controlled way of event-dependent sequencing and the handling of uncertainty during sequencing. A conventional finite state machine does provide the controlled way of event-dependent sequencing however it fails to capture the uncertainty involved. We view the tutoring process as an intelligently and dynamically planned information presentation cycle wherein certain level of imprecision is present. And hence a fuzzy finite state machine is suggested for tutor modeling. The proposed tutor model is novel because it is based on the observation of an experienced human tutor and captures tutorial strategies independent of the domain. As stated earlier, the studies indicate that the experienced human tutors provide the most effective form of instruction known. Our model tries to capture these characteristics. The testing results further strengthen this argument. The state to state transition is governed by the performance level of the learner. The repetition of the same lesson with different material is motivated from the principles of constructivist and cognitivist learning theories and is in line with the Blooms’ taxonomy [Bloom et al., 1965].

Presented below is the pointwise explanation regarding the proposed approach.
i. How are the human characteristics incorporated in the model?

Using a fuzzy finite state machine rather than a simple finite state automaton, the essential characteristics of a humanistic system are incorporated.

ii. How is the 'adaptivity' achieved?

The adaptivity is achieved through the performance input to the FFSM. As long as the performance remains at a particular level, the same lesson with different material is presented. The selection of lesson content is adaptive to the learners' knowledge level.

iii. Is the 'adaptivity' learner-centered?

As pointed out earlier, the transition from one state to the other as well as repeating the same state, both are decided on the basis of the learners' knowledge level and hence the model is learner-centric.

iv. How is the 'intelligence' incorporated?

Selecting the lesson content as per the learners' knowledge level, sequencing it to the learner at his own pace, testing the understanding and assimilation of the presented material are some of the 'intelligent' characteristics. These characteristics have been incorporated in the model through 'state' of FFSM. For every lesson aggregation, previous knowledge level of the learner is considered in each instance of the state.

v. Which of the learning theories advocate the model?

As mentioned earlier, the contractivistic learning theory and the cognitivistic learning theory advocate the model.

vi. How is the proposed model domain independent?

No assumption is made about the subject domain nor any subject dependent parameter is incorporated in the model. As long as the subject expert can properly
organize the content emphasizing and incorporating memorization, understanding and conception into the material, the given model can be used for any domain.

vii. Is the model technically feasible for web-based implementation?

We have demonstrated the feasibility of a FFSM using ANFIS which proves its technical feasibility. For web-based situation the requirements are little different as there can be several learners simultaneously. Such kind of functionalities catering to the requests of several clients are already handled in web servers through multithreading.

viii. What are the envisaged limitations?

The system-controlled nature of the model may constrain the learner. Similarly, the sequential nature of lesson presentation may not match with the learning style of every user.