Framework for Mining Hybrid Automata from a Constrained Machine Learning Architecture

Matthew Clark[1] and Kuldip S. Rattan[2]

[1]Principal Scientist, Galois Inc., Dayton, Ohio 45402
[2]Department of Electrical Engineering, Wright State University, Dayton, Ohio 45435

***Abstract*—As learning enabled cyber physical systems become more prolific, modeling for the purposes of verification and validation (V&V) becomes a primary barrier. A key challenge is to model cyber physical systems at the appropriate level of abstraction while maintaining a clear and understandable linkage between the human designer and the system under design. Fuzzy logic is a framework to synthesize linguistic, natural language requirements into machine learning models. It has been shown that fuzzy logic is a suitable method to learn the behavior of a nonlinear system when a model is not present. The implementation of a constrained fuzzy inference system presented in this paper is a specific class of machine learning architecture that provides a mechanism to relate a system model to human specified requirements. It is not at all dissimilar to more popular neural network architectures. The goal of this paper is to introduce a framework for constrained learning that enables both fast approximation of real valued systems while simultaneously enabling the modeling and analysis of safety properties. This paper focuses on a foundational constrained intelligent learning framework. The framework describes an alternative and constrained machine learning architecture capable of producing highly efficient structures that are also amenable to verification. This paper will focus on the efficiency of the computational structure while future papers will demonstrate the verifiability.**

I. Introduction

With the recent surge in the domain of deep learning with neural network architectures, new advances in data driven learning have emerged. However, many of the learning architectures pose a significant verification and reliability concern. The "hidden" layers of a deep neural network produce insufficient traceability between the resultant data driven model and the requirements by which the model must perform safely. At the same time, hybrid systems applicability has inspired a great deal of research in the modeling and verification of mixed signal systems [1] [2] [3]. In an effort to leverage hybrid system theory and verification with machine learning, the most promising work to date has been the approximation of a neural network as a hybrid system abstraction [4]. We propose that this framework is a direct translation to a more deterministic Hybrid System model.

Additionally, it is proposed that a constrained fuzzy logic architecture enables increased computational efficiency over traditional machine learning methods, provided the following conditions are satisfied: the fuzzification process uses a triangular membership function, the sum of the membership values over the universe of discourse at any instant for a control variable is equal to one, and the defuzzification method uses a modified center of area method [5] [6]. In this paper, we will demonstrate how constraining the learning algorithm architecture increases the computational efficiency while not sacrificing performance. Traditional machine learning architectures require all activation functions to be calculated at each time step, thereby, making the size of the network computationally prohibitive, especially on embedded devices. The number of activation function computations per cycle increases exponentially with the number of inputs and the number of layers. Our approach constrains the learning architecture such that the number of activation functions per layer and or input to only two, regardless of the number of total activation functions or layers. For example, for two inputs fuzzy system with 11 membership (activation) functions each, there are $11^2 = 121$ fuzzy rules (or equivalent computations). However, for four inputs fuzzy system with 11 membership functions each, there are $11^4 = 14,641$ fuzzy rules. Using Matlab on a 3.6 GHz CPU and 16 GB RAM, it takes 60 microeconds to produce an output for a 2-input systems. However, this times increases to 1200 microseconds for a 4-input system. This is not unique to fuzzy based machine learning algorithms. However, this is only true in the general case where little assumptions about the design architecture are made. KM-Logic (a Mamdani based computationally efficient, real-time recursive algorithm [7]), was developed to implement fuzzy inference systems. The algorithm constrains the general case to a specific fuzzy logic design. Within KM-Logic, input uncertainty is completely bounded within mathematically defined, adjacent triangular norm membership functions in which a maximum of two membership functions have non-zero membership value and the sum of the membership values is always equal to one. Thus, only two fuzzy rules from each input need to be considered to determine the firing values for each rule. Additionally, KM-Logic assumes weighted average defuzzification, providing an efficient representation that significantly reduces computation time. This significantly reduces the computational time. For example, using Matlab, it takes 30 microseconds to produce an output for a 2-input systems and this times increases almost linearly to only 120 microseconds for a 4-input system. Additionally, KM-Logic can be represented as a piecewise affine hybrid system, where each mode of the hybrid system is described as a piecewise n^{th} order polynomial, where n is governed by the number of inputs and or the number of layers of the network. This feature enables not only

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

increased computational efficiency but true transparency.

The paper is organized as follows: KM-Logic, a computationally efficient, real-time recursive algorithm to implement fuzzy inference systems for 1- and 2-input fuzzy system is described in section 2; hybrid system representation of KM-Logic is presented in section 3; the results based on Matlab simulations are given in section 4 and finally summary and conclusions are given in section 5.

II. KM-LOGIC

KM-Logic[1] (a Mamdani based computationally efficient, real-time recursive algorithm [7]), was developed to implement fuzzy inference systems in an efficient, verifiable way. KM-Logic is based on the fundamental theory from [5] and the foundational concepts from the book "NeuroFuzzy Adaptive Modeling and Control" where the concept of "knots" was introduced as a method to create constrained, fuzzy modes [6]. Fuzzy logic is a method of synthesizing natural language and intuitive requirements and specifications into an abstract model of desired behavior. Mamdami's [8] work introduced this control technology that Zadeh [9] [10] pioneered with his work in fuzzy sets. The KM-Logic algorithm constrains the general case fuzzy logic to a specific architectural design. Within KM-Logic, input uncertainty is completely bounded within mathematically defined adjacent triangular norm membership functions, in which a maximum of two membership functions have non-zero membership value and the sum of the membership values is always equal to one. The architecture of the KM-Logic algorithm constrains both the fuzzification and defuzzification processes. These constraints are summarized as follows.

Constraint 1: The input values from the sensors are considered crisp values. Therefore, fuzzification consists of matching the values to the input fuzzy membership functions over the domain of the input variables.
Constraint 2: The fuzzification process uses the triangular membership function.
Constraint 3: The width of a fuzzy set extends to the peak value of each adjacent fuzzy set and vice versa. Therefore, the sum of all membership over the universe of discourse at any instant for a control variable will always be equal to one. This constraint is referred to as **fuzzy partitioning** in this paper.
Constraint 4: The defuzzification method used is the modified center of area (weighted average) method.

In reality, there is not much difference between specific classes of Neural Network architectures and fuzzy logic. Notably, the relu activation function has been long since acknowledged as a practical constraint on neural network architectures for both speed [11] and verification reasons. The relu activation function is not unlike the fuzzy logic "triangular membership" function. In fact, KM-Logic leverages the implementation of adjacent triangular norm membership functions. Figure 1 a version of the KM-Logic architecture represented in the Neural Network visualization paradigm.

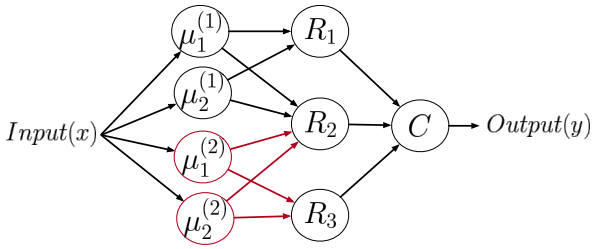

Fig. 1. Neural Network representation of KM-Logic.

The following sections provide examples of how KM-Logic is executed for 1 and 2 inputs. It should be noted that this method allows for the computation of n-input fuzzy systems by drastically reducing the computational complexity, reducing the number of rules to a fixed exponential growth dictated only by the number of inputs and not by number of membership functions as well.

A. 1-input, 1-output fuzzy system

If $X_1(j)$ are the membership functions in the $input_1$ (x_1) fuzzy set shown in Figure 2 and $U(o)$ are the membership functions in the output (u) fuzzy set shown in Figure 3, the linguistic rules describing a 1-inputs, 1-output (n=1) fuzzy system in natural language are of the form:

$$R_j\text{: } \textit{if } input_1 \ x_1 \textit{ is } X_1(j) \textit{ then output } u \textit{ is } U(o)$$

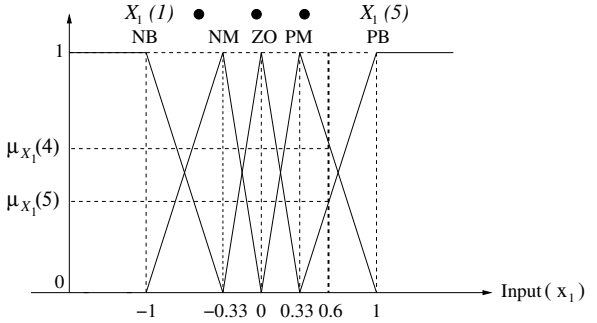

Fig. 2. Five triangular input membership function.

If both the input and output variables have five membership functions, the list of 5 rules $(R_1 \cdots R_5)$ in natural language may be written as:

$$\begin{aligned}
R_1: &\ \text{if } input_1\ x_1 \text{ is } X_1(1) \text{ then } output\ u \text{ is } U(1)\\
R_2: &\ \text{if } input_1\ x_1 \text{ is } X_1(2) \text{ then } output\ u \text{ is } U(2)\\
R_3: &\ \text{if } input_1\ x_1 \text{ is } X_1(3) \text{ then } output\ u \text{ is } U(3)\\
R_4: &\ \text{if } input_1\ x_1 \text{ is } X_1(4) \text{ then } output\ u \text{ is } U(4)\\
R_5: &\ \text{if } input_1\ x_1 \text{ is } X_1(5) \text{ then } output\ u \text{ is } U(5)
\end{aligned} \tag{1}$$

Note that the number of fuzzy rules depend only on the number of membership functions in the input and do not depend on the number of output membership functions. Therefore, the number of membership functions in output fuzzy set

[1]Provisional patent pending, Patent Application No: 62/313,183 and 62/362,267

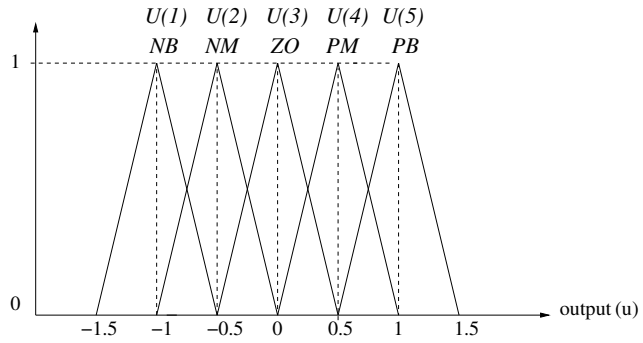

Fig. 3. Five triangular output membership functions.

do not have to be equal to the number of membership functions in input fuzzy sets. For example, for a 1-input fuzzy system with 5 fuzzy sets (NB, NM, Z0, PM and P) for input and output shown in figures 2 and 3, the rule-base can be written as

$$\begin{bmatrix} \text{Rule Location} \\ L_1 \\ L_2 \\ L_3 \\ L_4 \\ L_5 \end{bmatrix} \begin{bmatrix} x_1 \\ X_1(1) \\ X_1(2) \\ X_1(3) \\ X_1(4) \\ X_1(5) \end{bmatrix} \begin{bmatrix} u \\ U(1) \\ U(2) \\ U(3) \\ U(4) \\ U(5) \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ NB \\ NM \\ ZO \\ PM \\ PB \end{bmatrix} \begin{bmatrix} u \\ NB \\ NM \\ ZO \\ PM \\ PB \end{bmatrix} \quad (2)$$

Now, if the input x_1 lies within the center points of membership functions $X(2)$ and $X(3)$, then the two active rules from the list of rules given in equation (2) are given by

$$\begin{bmatrix} \text{Rule Location} \\ L_2 \\ L_3 \end{bmatrix} \begin{bmatrix} x_1 \\ X_1(2) \\ X_1(3) \end{bmatrix} \begin{bmatrix} u \\ U(2) \\ U(3) \end{bmatrix} \quad (3)$$

The location of the first active rule is $L_1 = 2$ and the location of the second active rule is $L_2 = 3$. For this example, if $u(2)$ and $u(3)$ are the center points of output fuzzy membership functions $U(2)$ and $U(3)$, respectively in equation (2), the active rule-list needed to find the output of the fuzzy system is represented as 2×1 matrix in equation (4) as

$$A_1 = \begin{bmatrix} u(2) \\ u(3) \end{bmatrix} = \begin{bmatrix} u(L_1) \\ u(L_2) \end{bmatrix} \quad (4)$$

where $L_1 = index_1$ is the value of the first active membership function and $L_2 = index_1 + 1$. Using weighted average defuzzification, the output of the 1-input, 1-output fuzzy system is given by

$$\begin{aligned} Output &= \mu_{x1}(index_1) \times u(L_1) + \mu_{x1}(index_1 + 1) \times u(L_2) \\ &= \begin{bmatrix} \mu_{x1}(index_1) & \mu_{x1}(index_1 + 1) \end{bmatrix} \times A_1 \quad (5) \\ &= f(1) \end{aligned}$$

where

$$f(i) = \mu_{x1}(index_1) \times u(L_i) + \mu_{x1}(index_1 + 1) \times u(L_i + 1) \quad (6)$$

is the output of the two adjacent active rules for $input_1$ (2^{nd} and 3^{rd} rules for this example).

1-input, 1-output fuzzy system example:

For an input value of $x_1 = 0.6$ in Figure 2, the first membership function that has non-zero value is $X_1(4)$. The fuzzification module determines the value of $index_1 = 4$. The membership values in membership functions $X_1(index_1)$ and $X_1(index_1 + 1)$ as $\mu_{x_1}(4) = 0.597$ and $\mu_{x_1}(5) = 0.403$, respectively. From the output fuzzy set, $u((L_1) =$ u(4) = 0.5 and $u((L_2) =$ u(5) = 1. Therefore, using equation (5), the output of the 1-input, 1-output fuzzy system is given by

$$\begin{aligned} \text{Output} &= \mu_{x_1}(index_1) \times u(L_1) + \mu_{x_1}(index_1 + 1) \times u(L_2) \\ &= \begin{bmatrix} 0.5 & 1 \end{bmatrix} \begin{bmatrix} 0.597 \\ 0.403 \end{bmatrix} \\ &= 0.7015 \quad (7) \end{aligned}$$

The control surface of one-input, one-output system is shown in Figure 4. It can be seen from this figure that the control surface is piecewise linear and the maximum and minimum output is 1 and -1, respectively. This is due to the fact that the center values of the PB and NB is restricted to 1 and -1, respectively. It can also be seen from this figure that if the input and fuzzy sets are equally spaced, the output is linear (output = input) for an input between -1 and 1, i.e, the gain of the fuzzy system is equal to 1.

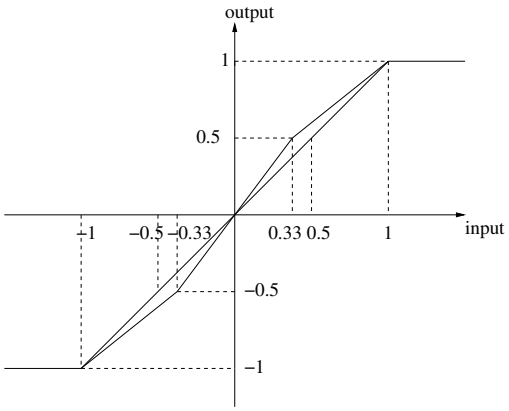

Fig. 4. Control surface of one-input, one-output fuzzy system.

B. 2-inputs, 1-output fuzzy system

For a 2-inputs 1-output fuzzy system (n=2), the linguistic rules can be described in natural language as

if $input_2$ (x_2) is $X_2(k)$ and $input_1$ (x_1) is $X_1(j)$ then output is $U(o)$

If each input fuzzy set has five membership functions, there are 25 rules ($R_1 \ \ldots \ R_{25}$). Note that the rules are listed in the order of the first membership function of $input_2$ and all membership functions of $input_1$ (5 rules in this case) followed by the second membership functions of $input_2$ and all membership functions of $input_1$ (5 rules in this case) and so on. Note again that the number of active rules (2^2 = 4 for two inputs) depend only on the number of inputs and not on the no of rules. Once the location of the first active rule is known, the location of the rest of the rules can be obtained

from the list of rules. The location of the first active rule, L_1, can be obtained using the following equations

$$L_1 = (index_1 + (index_2 - 1) \times (\text{no of membership functions in } input_1) \quad (8)$$

where $index_1$ is the value of the first active membership function in $input_1$ fuzzy set and $index_2$ is the value of the first active membership function in $input_2$ fuzzy set. Since two adjacent membership functions are active for each input, the location of the second active rule is always equal to $L_2 = L_1 + 1$. The location of the third active rule is given by $L_3 = L_1 +$ no of membership functions in $input_1$ and the location of the fourth active rules is equal to $L_4 = L_3 + 1$. Let $u(L_1)$, $u(L_2)$, $u(L_3)$ and $u(L_4)$ are the center points of output fuzzy sets for fuzzy rules R_{L_1}, R_{L_2}, R_{L_3} and R_{L_4}, respectively. and can be represented as 2×2 matrix in equation (9) as

$$\begin{bmatrix} u(L_1) & u(L_3) \\ u(L_2) & u(L_4) \end{bmatrix} = \begin{bmatrix} A_1(1) & A_2(1) \end{bmatrix} \quad (9)$$

where $A_1(1)$ is a 2×1 vector containing the center points of the output membership functions of the two active rules:"if $input_2$ is $X_2(index_2)$ and $input_1$ is $X_1(index_1)$ and if $input_2$ is $X_2(index_2)$ and $input_1$ is $X_1(index_1 + 1)$, and $A_2(1)$ is a 2×1 vector containing the center points of the output membership functions of the two active rules - "if $input_2$ is $X_2(index_2 + 1)$ and $input_1$ is $X_1(index_1)$ and if $input_2$ is $X_2(index_2 + 1)$ and $input_1$ is $X_1(index_1 + 1)$. Using weighted average defuzzification, the output of 2-inputs KM-Logic system is given by equation (10).

$$\begin{aligned} out(2) &= \mu_{x2}(index_2)\ (\mu_{x1}(index_1)\, u(L_1) + \mu_{x1}(index_1 + 1)\, u(L_2)) \\ &\quad + \mu_{x2}(index_2 + 1)\, [\mu_{x1}(index_1)\, u(L_3) + \mu_{x1}(index_1 + 1)\, u(L_4)] \\ &= \begin{bmatrix} \mu_{x2}(index_2) & \mu_{x2}(index_2 + 1) \end{bmatrix} \times \\ &\qquad \begin{bmatrix} \begin{bmatrix} \mu_{x1}(index_1) & \mu_{x1}(index_1 + 1) \end{bmatrix} \times A_1(1) \\ \begin{bmatrix} \mu_{x1}(index_1) & \mu_{x1}(index_1 + 1) \end{bmatrix} \times A_2(1) \end{bmatrix} \\ &= \begin{bmatrix} \mu_{x2}(index_2) & \mu_{x2}(index_2 + 1) \end{bmatrix} \begin{bmatrix} f(1) \\ f(3) \end{bmatrix} \end{aligned} \quad (10)$$

where $f(1)$ is the output using the membership values of two active membership functions of $input_1$ ($\mu_{x1}(index_1)$ and $\mu_{x1}(index_1 + 1)$ and the center points of the first two adjacent active output membership functions ($2^2/2$) obtained from the list of rules using equation (8) and $f(3)$ is the output using the membership values of $input_1$ ($\mu_{x1}(index_1)$ and $\mu_{x1}(index_1 + 1)$, and the center points of the next two adjacent active output membership functions obtained from the list of rules. Note that $f(1)$ and $f(3)$ are found using the membership functions of $input_1$ only using equation (6). These two output are then multiplied with the membership values of two adjacent active membership functions of $input_2$ (($\mu_{x2}(index_2)$ and $\mu_{x2}(index_2 + 1)$). This procedure has been implemented as a computationally efficient recursive algorithm for a n-input, 1-output system by selecting the center points of the output membership functions for the 2^n active rules from the rule list and finding the $2^n/2$ values of $f(i)$ and then multiplying them to the corresponding input membership functions as explained in equation (10).

Example of 2-input 1-output fuzzy system:

Now Consider a 2-input, 1-output fuzzy system with $input_1$ and $output$ fuzzy sets shown in Figures 2 and 3, and the $input_2$ fuzzy set shown in Figure **??**. Since each input variable has five membership functions, there are 25 rules ($R(1) \quad \ldots \quad R(25)$). Let the center points of the output membership functions for the 25 rules are given by the rule list given in equation (11) as

$$\begin{aligned} u = &[-1, -1, -1, -0.5, 0, -1, -1, -0.5, \underline{0, 0.5}, \\ &-1, -0.5, 0, \underline{0.5, 1}, -0.5, 0, 0.5, 1, 1, 0, 0.5, 1, 1, 1] \end{aligned} \quad (11)$$

For an input value of $x_1 = 0.6$ and $x_2 = -0.25$, $index_1 = 4$ and $index_2 = 2$ and the membership values of two active membership functions for each input are given by $\mu_{x_1}(index_1) = 0.597$, $\mu_{x_1}(index_1 + 1) = 0.403$, $\mu_{x_2}(index_2) = 0.5$, and $\mu_{x_2}(index_2+1) = 0.5$. The locations of the four active rules are given by $L(1) = 4+(2-1)*5 = 9$, $L(2) = 10$, $L(3) = 9 + 5 = 14$ and $L(4) = 15$. The center values of the output membership functions for the four active rules are underlined in equation (11) and are given by

$$\begin{bmatrix} A_1(1) & A_2(1) \end{bmatrix} = \begin{bmatrix} u(L_1) & u(L_3) \\ u(L_2) & u(L_4) \end{bmatrix} = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad (12)$$

The output of the 2-input fuzzy system can be calculated by first finding $f(1)$ and $f(3)$ (equation (13)) using the membership values of $input_1$ and the center points of the four output membership function given in equation (12) and then multiplying these values with the membership values of the active membership functions in $input_2$ fuzzy set given in equation (14).

$$\begin{aligned} f(1) &= \begin{bmatrix} \mu_{x1}(index_1) & \mu_{x1}(index_1 + 1) \end{bmatrix} \times A_1(1) \\ &= \begin{bmatrix} 0.597 & 0.403 \end{bmatrix} \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \\ &= 0.2015 \\ f(3) &= \begin{bmatrix} \mu_{x1}(index_1) & \mu_{x1}(index_1 + 1) \end{bmatrix} \times A_2(1) \\ &= \begin{bmatrix} 0.597 & 0.403 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \\ &= 0.7015 \end{aligned} \quad (13)$$

$$\begin{aligned} Output &= \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 0.2105 \\ 0.7015 \end{bmatrix} \\ &= 0.4515 \end{aligned} \quad (14)$$

The control surface of a 2-input, 1-output fuzzy system described above is shown in Figure 5. It can be seen from this figure that the control surface is nonlinear at the corners.marked by mode numbers 1, 2, 5, 12, 15, 16. This is due to the fact that the output of the fuzzy system has been restricted to a maximum and minimum values of 1 and -1, respectively. Since four rules are active for any mode, the center points of the output membership functions of all four

rules at the corner are -1 or 1, respectively for any input combinations in modes 1 and 16. Therefore, the output of the fuzzy system for any combination of $input_1$ between -1 and -0.33, and $input_2$ between -1 and -0.5 is equal to -1. Similarly, the output of the fuzzy system for any combination of $input_1$ between 0.33 and 1 and $input_2$ between 0.5 and 1 is equal to 1. The output of the fuzzy system for modes 2, 5, 12 and 14 is nonlinear because of the output of the three rules are restricted to -1 or 1.

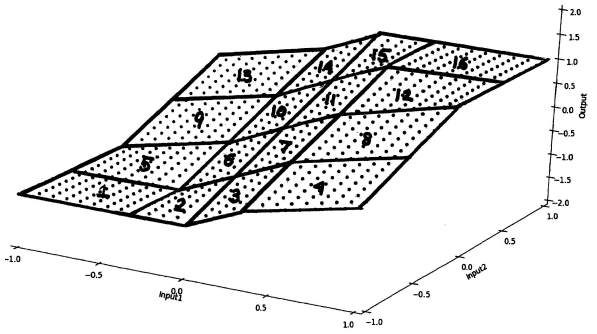

Fig. 5. Nonlinear surface of 2-input fuzzy system.

III. Hybrid System representation of KM-Logic

A hybrid system is a system with dynamics that are dependent upon both continuous and discrete state variables. [12]. A hybrid automaton is a modeling formalism for hybrid systems which is defined in [13] by the octuple: $H = (Q, X, f, Init, D, E, G, R)$, where

- Q is a finite set of discrete variables;
- X is a finite set of continuous variables;
- $f : Q \times X \to X$ is a vector field defining the dynamics of the continuous variables;
- $Init \subset Q \times X$ is the set of initial states;
- $D : Q \to P(X)$ defines the domain of the discrete modes
- $E \subset Q \times Q$ is a set of edges;
- $G : E \to P(X)$ defines guard conditions for discrete transitions;
- $R : E \times X \to P(X)$ is a reset map defining discontinuities in the continuous state of the system during discrete transitions.

The state of the system is the collection of continuous and discrete states and will be denoted by $(x, q) \in X \times Q$. The formal hybrid automaton which defines the thermostat system is

- $Q = \{q_{\text{on}}, q_{\text{off}}\}$;
- $X = \mathbb{R}$;
- $f(q_{\text{on}}, x) = c_1 x$,
  $f(q_{\text{off}}, x) = -c_2 x$;
- $Init = q_{\text{off}} \times \{x \in X : x \leq T_{\text{high}}\}$;
- $D(q_{\text{on}}) = \{x \in X : x \leq T_{\text{high}}\}$,
  $D(q_{\text{off}}) = \{x \in X : x \leq T_{\text{low}}\}$;
- $E = \{(q_{\text{on}}, q_{\text{off}}), (q_{\text{off}}, q_{\text{on}})\}$;
- $G(q_{\text{on}}, q_{\text{off}}) = \{x \in \mathbb{R} : x \geq T_{\text{high}}\}$,
  $G(q_{\text{off}}, q_{\text{on}}) = \{x \in \mathbb{R} : x \leq T_{\text{low}}\}$;

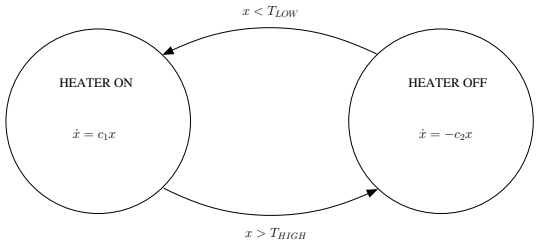

Fig. 6. Illustration of simple heater/thermostat hybrid automaton.

- $R(q_{\text{on}}, q_{\text{off}}, x) = R(q_{\text{off}}, q_{\text{on}}, x) = x$.

A common example of a hybrid system is a simple thermostat as shown in figure 6 where the control of the heater is represented by discrete states of (Heater On) and (Heater Off). The dynamics of the heat in the room is represented by the differential equations in each state.

A. Hybrid System representation of a 1-input KM-Logic system

The piecewise linear surface shown in Figure 4 has 4 (number of membership functions in ($input_1$ - 1)) linear pieces called the modes. The output of mode m can be represented by $Output(m) = K_1(m)\, x_1 + Const(m)$, where $K_1(m)$ and $Const(m)$ are the gain and the constant terms, respectively of mode m. In this example, mode 1 ($m = 1$) represents input values between -1 and -0.33, mode 2 ($m = 2$) represents input values between -0.33 and 0, mode 3 ($m = 3$) represents input values between 0 and 0.33 and finally mode 4 ($m = 4$) represents input values between 0.33 and 1. The expressions for $K_1(m)$ and $Const(m)$ are derived by substituting the expression for $\mu_{x_1}(m)$ and $\mu_{x_1}(m+1)$ into equation (7) as

$$\begin{aligned} Out(m) &= \frac{x_1(m+1) - x_1}{x_1(m+1) - x_1(m)} u(m) + \frac{x_1 - x_1(m)}{x_1(m+1) - x_1(m)} u(m+1) \\ &= \frac{1}{x_1(m+1) - x_1(m)} [(u(m+1-u(m))x_1 \\ &\qquad + (x_1(m+1)u(m) - x_1(m)u(m+1))] \end{aligned}$$

Therefore,

$$\begin{aligned} K_1(m) &= \frac{u(m+1) - u(m)}{Den} \\ &= \frac{1}{Den} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} u(m) \\ u(m+1) \end{bmatrix} \end{aligned} \tag{15}$$

and

$$\begin{aligned} Const(m) &= \frac{x_1(m+1)\, u(m) - x_1(m)\, u(m+1)}{Den} \\ &= \frac{1}{Den} \begin{bmatrix} x_1(m+1) & -x_1(m) \end{bmatrix} \begin{bmatrix} u(m) \\ u(m+1) \end{bmatrix} \end{aligned} \tag{16}$$

where $Den = x_1(m+1) - x_1(m)$. Note that $K_1(m)$ can be obtained directly from equation (5) by replacing the membership values by -1 and dividing the result by the difference of the center values of input membership functions $X_1(m+1)$ and $X_1(m)$. Also, the value of the $Const(m)$ term can be obtained directly from equation (5) by replacing the membership value $\mu_{x_1}(m)$ by the center value of input membership function $X_1(m+1)$ and the membership value $\mu_{x_1}(m+1)$ by -1 $times$ the center value of the input membership function $X_1(m)$ and dividing the result by the Den. The hybrid representation of a one-input one-output system is shown in Figure 7.

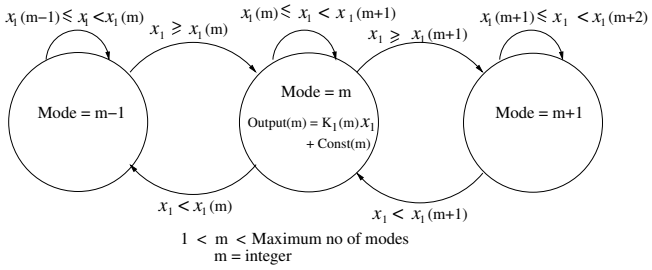

Fig. 7. Piecewise linear representation of one-input, one-output fuzzy system.

For example, If $x_1 = 0.6$, $m = 4$ and the values of $K_1(4)$ and $Const(4)$ can be calculated as

$$\begin{aligned} K_1(4) &= \frac{1}{0.67}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}0.5\\ 1\end{bmatrix} \\ &= 0.74626 \\ Const(4) &= \frac{1}{0.67}\begin{bmatrix}1 & 0.33\end{bmatrix}\begin{bmatrix}0.5\\ 1\end{bmatrix} \\ &= 0.25373 \end{aligned}$$

These values can be used to find the output of the system for the values of input between o.33 and 1. Therefore, for $x_1 = 0.6$, the output of the fuzzy system is given by

$$\begin{aligned} Output &= 0.74626 \times 0.6 + 0.25373 \\ &= 0.7015 \end{aligned}$$

Note that this value is exactly the same as the value obtained earlier in equation (14).

B. Hybrid System implementation of a 2-input KM-Logic system

An alternate way to use the recursive algorithm explained above is to generate piecewise polynomial gains governed by the input modes as described below to create a hardcoded lookup table of coefficients. It can be seen from Figure 5 that there are 16 modes ($m_1 \times m_2$), where m_1 = (no of fuzzy membership functions in ($input_1$ - 1)) = 4 and m_2 = (no of fuzzy membership functions in ($input_2$ - 1)) = 4. The output of the 2-input fuzzy system for mode m can be written as $Output(m) = K_1(m)\,x_1 + K_2(m)\,x_2 + K_{12}(m)\,x_1\,x_2 + Const(m)$, where $K_1(m)$ is the gain for $input_1$, $K_2(m)$ is the gain for $input_2$, $K_{12}(m)$ is the gain of the nonlinear term and $Const(m)$ is the constant term of mode m. The expressions for gain and constant terms are derived by substituting the expression for $\mu_{x_1}(index_1)$, $\mu_{x_1}(index_1+1)$, $\mu_{x_2}(index_2)$, $\mu_{x_2}(index_2+1)$, $u(L_1)$, $u(L_2)$, $u(L_3)$ and $u(L_4)$ into equation (10). After simplification gives,

$$\begin{aligned} K_1(m) =& \frac{1}{Den}\begin{bmatrix}x_2(index_2+1) & -x_2(index_2)\end{bmatrix}\begin{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}u(L_1)\\ u(L_2)\end{bmatrix}\\ \\ \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}u(L_3)\\ u(L_4)\end{bmatrix}\end{bmatrix} \\ =& \frac{1}{Den}\Big[x_2(index_2+1)\times\Big(u(L_2)-u(L_1)\Big) \\ & - x_2(index_2)\times\Big(u(L_4)-u(L_3)\Big)\Big] \end{aligned}$$

$$\begin{aligned} K_2(m) =& \frac{1}{Den}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}\begin{bmatrix}x_1(index_1+1) & -x_1(index_1)\end{bmatrix}\begin{bmatrix}u(L_1)\\ u(L_2)\end{bmatrix}\\ \\ \begin{bmatrix}x_1(index_1+1) & -x_1(index_1)\end{bmatrix}\begin{bmatrix}u(L_3)\\ u(L_4)\end{bmatrix}\end{bmatrix} \\ =& \frac{1}{Den}\Big[x_1(index_1+1)\times\Big(u(L_3))-u(L_1)\Big) \\ & - x_1(index_1)\times\Big(u(L_4)-u(L_2)\Big)\Big] \end{aligned}$$

$$\begin{aligned} K_{12}(m) =& \frac{1}{Den}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}u(L_1)\\ u(L_2)\end{bmatrix}\\ \\ \begin{bmatrix}-1 & -1\end{bmatrix}\begin{bmatrix}u(L_3)\\ u(L_4)\end{bmatrix}\end{bmatrix} \\ =& \frac{1}{Den}\Big[\Big(u(L_4)-u(L_3)\Big)-\Big(u(L_2)-u(L_1)\Big)\Big] \end{aligned}$$

and

$$\begin{aligned} Const(m) =& \frac{1}{Den}\begin{bmatrix}x_2(index_2+1) & -x_2(index_2)\end{bmatrix}\times \\ & \begin{bmatrix}\begin{bmatrix}x_1(index_1+1) & -x_1(index_1)\end{bmatrix}\begin{bmatrix}u(L_1)\\ u(L_2)\end{bmatrix}\\ \\ \begin{bmatrix}x_1(index_1+1) & -x_1(index_1)\end{bmatrix}\begin{bmatrix}u(L_3)\\ u(L_4)\end{bmatrix}\end{bmatrix} \\ =& \frac{1}{Den}\begin{bmatrix}x_1(index_1+1)(x_2(index_2+1)u(L_1)-x_2(index_2)u(L_3))\\ \\ -x_1(index_1)l(x_2(index_2+1)u(L_2)-x_2(index_2)u(L_4))\end{bmatrix} \end{aligned}$$

where

$$Den = [(x_2(index_2+1) - x_2(index_2)] \times [(x_1(index_1+1) - x_2(index_1)]\,.$$

If $input_1$ lies between 0.33 and 1 and $input_2$ between -0.5 and 0, the output is defined by mode $m = 8$. The values of $K_1(8)$, $K_2(8)$, $K_{12}(8)$, and $Const(8)$ can be calculated as

$$K_1(8) = \frac{1}{0.335}\begin{bmatrix}0 & 0.5\end{bmatrix}\begin{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}0\\ 0.5\end{bmatrix}\\ \\ \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}0.5\\ 1\end{bmatrix}\end{bmatrix}$$

$$= 0.74626$$

$$K_2(8) = \frac{1}{0.335}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}\begin{bmatrix}1 & 0.5\end{bmatrix}\begin{bmatrix}0\\0.5\end{bmatrix}\\ \begin{bmatrix}1 & 0.5\end{bmatrix}\begin{bmatrix}0.5\\1\end{bmatrix}\end{bmatrix}$$

$$= 1.0$$

$$K_{12}(8) = \frac{1}{0.335}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}0\\0.5\end{bmatrix}\\ \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}0.5\\1\end{bmatrix}\end{bmatrix}$$

$$= 0$$

$$Const(8) = \frac{1}{0.335}\begin{bmatrix}0 & 0.5\end{bmatrix}\begin{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}0\\0.5\end{bmatrix}\\ \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}0.5\\1\end{bmatrix}\end{bmatrix}$$

$$= 0.25374$$

These values can be used to find the output of the system for $0.33 \leq x_1 \leq 1$ and $-0.5 \leq x_2 \leq 0$. Therefore, for $x_1 = 0.6$ and $x_2 = -0.25$, the output of the system is given by

$$Output = 0.74626 \times 0.6 + 1 \times (-0.25) + 0 \times (0.6) \times (0.25) + 0.25374$$
$$= 0.4515$$

Note that this value is exactly the same as the value obtained earlier in equation (14).

IV. Simulation Results

A detailed KM-Logic architecture is described in this paper. This architecture is suitable for classification and machine learning with increased performance and computational speed. The algorithm increases the computational efficiency while not sacrificing performance. Traditional machine learning architectures require all activation functions to be calculated at each time step, thereby, making the size of the network computationally prohibitive, especially on embedded devices.

To test the computational efficiency of KM-Logic, a fuzzy logic system was implemented using KM-Logic and the traditional methods. The following hypotheses are tested:

1) Using KM-Logic, the computational time to implement a fuzzy system remains almost the same as the number of membership functions are increased to fuzzify inputs. However, in the case of traditional methods, the computational time increases substantially as the number of membership functions are increased.
2) The implementation time increases linearly with the number of inputs in the case of KM-Logic, whereas this time increases almost exponentially for traditional implementations.
3) For a linear fuzzy system, only 1-mode is needed to implement this system, i.e., only 2 membership functions are needed for each input.

To test these hypotheses, first a fuzzy system is obtained for the function $f(x) = x^4$ by learning the rule-base for one, two, three and four inputs. First, the output data is obtained over the input domain of -1 to 1 in step of 1 $msec$. Using this input/output data, a learning algorithm is used to obtain the rule-base. The execution times for each input is shown in the figure 8. It can be seen from this figure that KM-Logic only takes 0.097 milliseconds to perform each computation. The computation time only increases to 0.105 milliseconds when the number of membership functions increases to 9 for each input. On the other hand, for a traditional implementation, the computation time is almost 8 orders of magnitude slower. It should be noted that most of the constraints are still enforced for both implementations. Specifically, adjacent triangular membership functions. For a more general Fuzzy Logic design, the computational cost would be significantly higher. Also, to achieve the same performance, most classical neural networks use 128x128 activation functions at minimum; which further increase the computational cost unnecessarily.

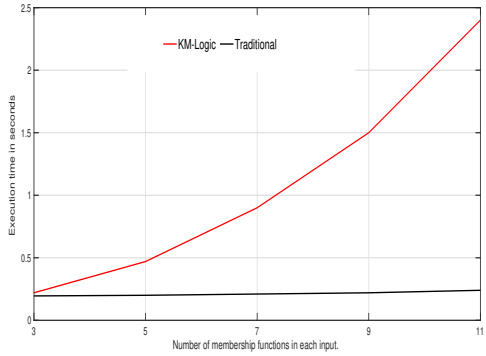

Fig. 8. Execution times of fuzzy systems with 4 inputs, comparing KM-Logic to Traditional Fuzzy Logic

The execution time for both the traditional implementation and KM-Logic was also obtained using Matlab for 4-input systems with 3, 5, 7 and 9 membership functions for each input. and the results are shown in figure 9. It can be seen from this figure that the execution time increases linearly with the number of inputs in the case of KM-Logic, whereas this time increases almost exponentially for a traditional implementation.

Leveraging the direct Hybrid System equivalance of the KM-Logic framework, The function $f(x) = x^4$ can actually be modeled with only one membership function per input if each input is a copy of x. In other words, it is possible, in many cases to obtain an exact or near exact approximation of the modeled function through direct mathematical translation.

V. Summary and Conclusions

One of the drawbacks of fuzzy inference systems and neural networks is the computation time it takes to combine the membership values of every input to form the firing value of each rule. For a given number of membership functions in each input fuzzy set, the number of fuzzy rules increase

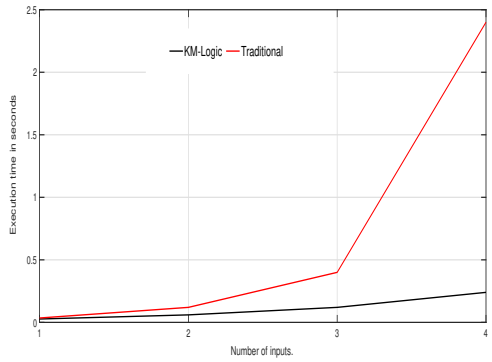

Fig. 9. Exponential vs Linear computation of classical to KM-Logic Fuzzy framework

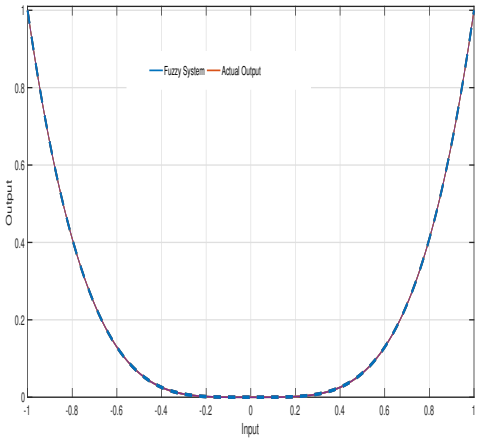

Fig. 10. Comparison of the KM-Logic Approximation vs the actual output

geometrically with the number of inputs. This geometric progression of the number of rules becomes an obstacle for practical application of multiple-input systems because of the time it takes to implement these systems. For a four input fuzzy system with 3 membership function in each input, using Matlab it took 0.11 microseconds to implement the fuzzy system. However, this time increased to 750 microseconds if each input is fuzzified using 9 membership function. To overcome this geometric progression, a novel recursive algorithm, KM-Logic, was developed. Simulation results have shown that this method significantly reduces the implementation time. Using Matlab, for a 4-input fuzzy system with 3 membership function each, it takes 0.097 milliseconds to implement the fuzzy system and this time increases to only 0.105 milliseconds if each input is fuzzified using 9 membership functions. Also, it was shown that the implementation time increases linearly as the number of inputs are increased using KM-Logic whereas in the case of traditional implementations, the computational times increases exponentially.

REFERENCES

[1] Matthew Clark, Xenofon Koutsoukos, Joseph Porter, Ratnesh Kumar, George Pappas, Oleg Sokolsky, Insup Lee, and Lee Pike. A study on run time assurance for complex cyber physical systems. Technical report, Air Force Research Lab, Wright-Patterson AFB ,Aerospace Systems Directorate, ohio, 2013.

[2] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1992.

[3] Cory Firmin Snyder. Provable run time safety assurance for a non-linear system. 2013.

[4] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178. ACM, 2019.

[5] Thomas Brehm and Kuldip S Rattan. The classical controller: a special case of the fuzzy logic control. In *Proceedings of the 33rd IEEE Decision and Control Conference*, pages 4128–4129, 1994.

[6] Martin Brown and Chris Harris. *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall International (UK) Ltd., GBR, 1995.

[7] Matthew Clark, Kuldip Rattan, Nicholas Ernest, Tim Arnett, and Kelly Cohen. *Verification and Validation of a Genetic Fuzzy Tree for UCAV Control*. 2^{nd} Edition Intelligent System Book, AIAA, 2017.

[8] E. H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. In *Proc. IEE 121*, volume 12, pages 1585–1588, 1974.

[9] L. A. Zadeh. Fuzzy sets. In *Information and Control*, volume 8, pages 338–353, 1965.

[10] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. In *IEEE Trans. of Systems, Man, Cybernatics*, volume 3, pages 28–44, 1973.

[11] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[12] Thomas A Henzinger. *The theory of hybrid automata*. Springer, 2000.

[13] John Lygeros, Karl Henrik Johansson, Slobodan N Simic, Jun Zhang, and Shankar Sastry. Continuity and invariance in hybrid automata. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 340–345. IEEE, 2001.