Symbolic Verification of Fuzzy Logic Models

Siang Zhao*†, Zhongyang Li*†, Zhenbang Chen*†§, Ji Wang*‡
*College of Computer, National University of Defense Technology, China
†Key Laboratory of Software Engineering for Complex Systems, National University of Defense Technology, China
‡State Key Laboratory of High Performance Computing, National University of Defense Technology, China
{zhaosiang16, zbchen, wj}@nudt.edu.cn
§Corresponding author

***Abstract*—Fuzzy logic is widely applied in various applications. However, verifying the correctness of fuzzy logic models can be difficult. This extended abstract presents our ongoing work on verifying fuzzy logic models. We treat a fuzzy logic model as a program and propose a verification method based on symbolic execution for fuzzy logic models. We have developed and implemented the environment models for the common functions and the inference rules in fuzzy logic models. Our preliminary evaluation shows the potential of our verification method.**

***Index Terms*—fuzzy logic model, verification, symbolic execution, SMT**

I. Introduction

Fuzzy logic [1] has a long history of development [2] and has been widely applied in many areas for controlling and decision-making. However, it is challenging to ensure the correctness and reliability of fuzzy models or fuzzy model-based systems for safety-critical applications or systems employing fuzzy logic models.

A fuzzy logic model $\mathcal{M}$ takes a set of inputs (denoted by $\mathcal{I}$) and produces an output (denoted by $\mathcal{O}$). In general, $\mathcal{M} = (\mathcal{I}, \mathcal{O}, \mathcal{F}, \mathcal{R}, \mathcal{D})$ consists of three computation components: fuzzification (denoted by $\mathcal{F}$) maps inputs in $\mathcal{I}$ to the fuzzy sets by a set of fuzzification functions. Inference (denoted by $\mathcal{R}$) contains a set of rules, each of which specifies the logical relation between the fuzzy values of inputs and output. Defuzzification (denoted by $\mathcal{D}$) defines the functions for transforming a fuzzy value to a continuous output value. Figure 1 illustrates a fuzzy logic model for controlling the boiling time of eggs.

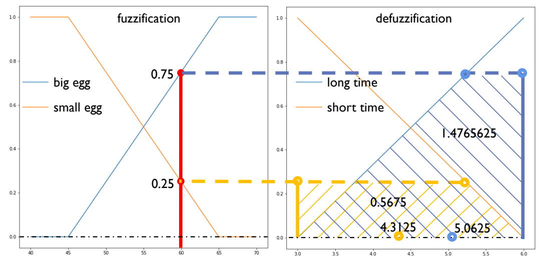

Fig. 1. A fuzzy model example

This model takes one input (*i.e.*, the egg's weight) and produces the boiling time as the output. There are two fuzzification functions for big and small eggs and two defuzzification functions for long and short boiling times, respectively. The fuzzy values of long or short time equal the values of big or small eggs. Suppose the egg's weight is 60 grams. The degrees of membership for big and small egg functions are 0.75 and 0.25, respectively. Therefore, the fuzzy values of membership for long and short periods are 0.75 and 0.25. Suppose that we use the centroid method [3] for defuzzification, which computes the output as follows:

$$output = \frac{c_s \times S_s + c_l \times S_l}{S_s + S_l}$$

where c_s and c_l represent the abscissa of the centroids of the shaded areas of yellow and blue, respectively; S_s and S_l represent the areas of the yellow and blue rectangles, respectively. For this example, c_s, c_l, S_s and S_l are 4.3125, 5.0625, 0.5675 and 1.4765625, respectively. Therefore, the output value is 4.854. The functions and rules would be more complex for real-world fuzzy logic models.

A fuzzy logic model is usually designed manually by its designer, and the quality of the model depends on the designer's expertise. Testing or simulation [4] is the main method for ensuring the correctness of the model. However, verifying the correctness of fuzzy logic models is challenging. For example, back to the model in Figure 1, no matter how small the egg is, it would be boiled for at least 3.75 minutes to ensure it is cooked. However, for more complex models, it is hard to manually ensure the model's reliability. Very little work exists on verifying fuzzy logic models [5] [6]. Arnett *et al.* [7] propose to use Z3 [8] for verifying fuzzy logic models by abstracting the model into a polynomial one. However, their approach has two limitations: 1) the abstraction method needs to be more general; 2) the scalability is a problem due to Z3's weakness in solving floating-point constraints.

This extended abstract presents our recent progress in verifying fuzzy logic models. Our key idea is to treat a fuzzy logic model as a program and apply program verification techniques for verification. Our key observation is that *there are no input-related loops inside the programs implementing fuzzy logic models*. Therefore, we propose using symbolic execution [9] to systematically explore the path space of a model's program and verify the correctness of the model. We implemented our method in a prototype and conducted preliminary experiments on representative fuzzy logic models. The experimental results show the potential of our method's effectiveness.

II. Verification Framework

Figure 2 shows our verification framework. The verifier takes two inputs: a fuzzy logic model and the properties. The

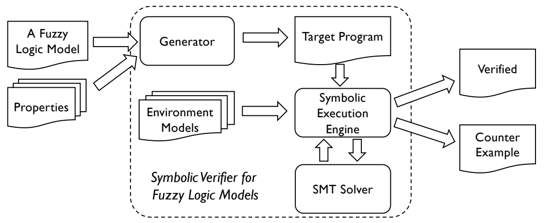

Fig. 2. Our Verification Framework

properties are relations between the model's inputs and output, which can be specified as the pre and post-conditions [10] in program verification. Quantifier-free first-order logic formulas of input and output variables can specify the properties. Our verifier's outputs are the verification results, *i.e.*, the model is verified to satisfy the properties or counter-examples exist. The model's designer can use the counter-examples to replay and fix the model. Next, we explain the key components of our framework.

Generator & Environment modeling. These two components are related. Although each original fuzzy logic model is implemented and executed with a fuzzy model library, we developed the environment models for verification. First, we can improve the scalability of verification by pruning redundant cases; Second, customization makes it easy to make the approximations for non-linear $\mathcal{D}$. Until now, our environment models support the most common fuzzification and defuzzification functions and the functions for all the logic operators of fuzzy inference rules. Therefore, based on environment models, the generator generates target programs by invoking the predefined functions in the environment models. It also converts the properties into assertions in the target program, which will be verified later by symbolic execution.

Symbolic execution engine. We employ symbolic execution to systematically explore the path space of the model's target program. We use path conditions and symbolic values along the path to check the properties. More specifically, we add the pre-conditions of the inputs to each path's path condition at the beginning; When a path terminates, we use its path condition PC to check the validity of the post-condition. For example, if the post-condition is C, we query the SMT solver whether the formula $PC \wedge \neg C$ is satisfiable. If it is satisfiable, we find a counter-example, *i.e.*, the solution of $PC \wedge \neg C$; Otherwise, the path satisfies C, and symbolic execution continues to explore and check other paths. If all the paths are explored, and no counter-example is found, the model is verified to satisfy the property.

SMT Solver. The SMT solver component is also crucial for our verification framework. Intensive (and non-linear) floating-point computations are inside the fuzzification and defuzzification functions. Besides, many elementary mathematical functions may also be used. Hence, employing the bit-vector SMT theories (e.g., BVFP) for a precise representation is not feasible. In early experiments, we found that verification with the Z3 takes an unacceptably long time, even if the model is simple. In practice, we consider each input as a real number and employ real arithmetic SMT theory for program representation. Hence, the employed SMT solver in our framework is a real arithmetic solver. Although real arithmetic solvers cannot support some operations, e.g., bit and logic operations, these operations are rare for the functions in fuzzy logic models.

When users provide the model and the properties, our framework can automatically verify the model with respect to the properties. Note that our verification framework does not precisely encode floating-point operations due to using real arithmetic SMT solving. Hence, the counter-example may be a false positive and needs to be confirmed by concrete execution.

III. Preliminary Evaluation

We have implemented our framework in a prototype. The target program and the environment models are Python programs. We have designed and implemented a dynamic symbolic execution [11] engine for Python programs. We chose DSE because, for interpreted languages, DSE is the only option. The SMT solver is dReal [12], which supports real arithmetic SMT solving of non-linear constraints and many elementary mathematical functions (*e.g.*, power functions, and trigonometric functions).

We have conducted a preliminary evaluation of twelve fuzzy logic models. Many of them are examples from different fuzzy logic libraries. There are nine atomic models and three composite models. The number of inputs is usually one or two. The largest number of inputs is six, and the model is composite. For atomic models, the number of outputs is one. Only two composite models have two outputs. The number of inference rules ranges from 2 to 49. Our prototype can verify all the models, and the properties are given from our understanding of the models (*i.e.*, the models should satisfy the property). The verification time of ten models is below half a minute. For the other two models (*i.e.*, the ones with the largest number of inputs and rules, respectively), our prototype needs more than 7 hours for verification.

IV. Related Work and Future Plans

Very little work exists on verifying fuzzy logic models. Besides the work in [7], Ding *et al.* [13] convert a fuzzy logic model into a hybrid automaton and use the existing model checking tool to verify the model's stability. As far as we know, our framework is the first general verification framework for fuzzy logic models and the first work on verifying fuzzy logic models on the code level.

There are the following aspects for the next step: 1) more extensive evaluations on more fuzzy logic models; 2) investigation of the scalable verification method for composite models; 3) our framework is limited to continuous functions, which is also a challenge for verification; 4) tool development to make it easier to use.

Acknowledgment

This research was supported by National Key R&D Program of China (No. 2022YFB4501903) and the NSFC Programs (No. 62172429 and 62032024).

References

[1] L. A. Zadeh, "Fuzzy sets," in *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*. World Scientific, 1996, pp. 394–432.

[2] S. Chiu, "Using fuzzy logic in control applications: beyond fuzzy pid control," *IEEE Control Syst*, vol. 18, no. 5, pp. 100–104, 1998.

[3] M. H. Center, "Defuzzification methods in mathworks," https://ww2.mathworks.cn/help/fuzzy/defuzzification-methods.html.

[4] K. S. Rattan, M. A. Clark, and J. A. Hoffman, "Design and analysis of a multistage fuzzy pid controller," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 5726–5731.

[5] H. Fang, H. Zhu, and J. He, "Smt-based symbolic encoding and formal analysis of hml models," *Mobile Networks and Applications*, vol. 21, pp. 35–52, 2016.

[6] P. J. Prieto-Entenza, N. R. Cazarez-Castro, L. T. Aguilar, S. L. Cardenas-Maciel, and J. A. Lopez-Renteria, "A lyapunov analysis for mamdani type fuzzy-based sliding mode control," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 8, pp. 1887–1895, 2019.

[7] T. J. Arnett, B. Cook, M. Clark, and K. Rattan, "Fuzzy logic controller stability analysis using a satisfiability modulo theories approach," in *AIAA-17*, 2017, p. 1773.

[8] L. d. Moura and N. Bjørner, "Z3: An efficient smt solver," in *TACAS*. Springer, 2008, pp. 337–340.

[9] J. C. King, "Symbolic execution and program testing," *Communications of the ACM*, vol. 19, no. 7, pp. 385–394, 1976.

[10] C. Morgan, *Programming from specifications*. Prentice-Hall, Inc., 1990.

[11] M. Irlbeck *et al.*, "Deconstructing dynamic symbolic execution," *MSR-TR*, vol. 40, p. 26, 2015.

[12] S. Gao, S. Kong, and E. M. Clarke, "dreal: An smt solver for nonlinear theories over the reals," in *CADE*. Springer, 2013, pp. 208–214.

[13] Z. Ding, Y. Zhou, and M. Zhou, "Stability analysis of switched fuzzy systems via model checking," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 6, pp. 1503–1514, 2014.