IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 25, 2019, accepted September 10, 2019, date of publication September 20, 2019, date of current version October 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942762

LTL Model Checking Based on Binary Classification of Machine Learning

WEIJUN ZHU[1], HUANMEI WU[2], AND MIAOLEI DENG[3]
[1]School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China
[2]School of Informatics & Computing, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202, USA
[3]School of Informatics, Henan University of Technology, Zhengzhou 450001, China

Corresponding author: Weijun Zhu (zhuweijun76@163.com)

This work was supported by the National Natural Science Foundation of China under Grant U1204608.

ABSTRACT Linear Temporal Logic (LTL) Model Checking (MC) has been applied to many fields. However, the state explosion problem and the exponentially computational complexity restrict the further applications of LTL model checking. A lot of approaches have been presented to address these problems. And they work well. However, the essential issue has not been resolved due to the limitation of inherent complexity of the problem. As a result, the running time of LTL model checking algorithms will be inacceptable if a LTL formula is too long. To this end, this study tries to seek an acceptable approximate solution for LTL model checking by introducing the Machine Learning (ML) technique. And a method for predicting LTL model checking results is proposed, using the several ML algorithms including Boosted Tree (BT), Random Forest (RF), Decision tree (DT) or Logistic Regression (LR), respectively. First, for a number of Kripke structures and LTL formulas, a data set A containing model checking results is obtained, using one of the existing LTL model checking algorithm. Second, the LTL model checking problem can be induced to a binary classification problem of machine learning. In other words, some records in A form a training set for the given machine learning algorithm, where formulas and kripke structures are the two features, and model checking results are the one label. On the basis of it, a ML model M is obtained to predict the results of LTL model checking. As a result, an approximate LTL model checking technique occurs. The experiments show that the new method has the similar max accuracy with the state of the art algorithm in the classical LTL model checking technique, while the average efficiency of the former method is at most 6.3 million times higher than that of the latter algorithms, if the length of each of LTL formulas equals to 500. These results indicate that the new method can quickly and accurately determine LTL model checking result for a given Kripke structure and a given long LTL formula, since the new method avoids the famous state explosion problem.

INDEX TERMS Machine learning, model checking, linear temporal logic, binary classification.

I. INTRODUCTION

Model checking was presented by Turing Award winner Prof. Clarke *et al.* [1]. Up to now, model checking has been applied to many fields of computer science and systems engineering, such as microprocessor design & computer chip design [2], security protocols [3] and malware detection [4], and this technique has been used by some leading IT companies, including INTEL and IBM [5].

What a system is in the field of systems engineering? According to NASA's definition [6], ''the combination of elements that function together to produce the capability to meet a need'', and ''the elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose''. In fact, model checking provides an effective way for such a complex system, to formally determine whether a system has the capability to meet a given need or not. Thus, model checking has been applied to many aspects of systems engineering, such as spacecraft design [7], robotics [8], [9], human-automation interaction [10], [11] and group behavior interactions [12]. For many cases of system developments, not only early phases but also later ones need model checking. For examples, Microsoft used this formal verification technique to check real code, and the model checker found some distinct errors and bugs in codes [13].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

In general, temporal logic formulas are used for formal specifications of elements (hardware or software) or systems, while automata, Kripke structures, Petri nets or Process algebra are employed to describe models of elements or systems. On the basis of it, the model checking algorithms can automatically verify whether the systematic model satisfy the required property and specification or not, during systems analysis, requirements analysis, and systems design. For a system, the result of model checking will be ''true/yes'', if the Kripke structure satisfies the formula, i.e., the systematic model satisfy the required property. Otherwise, the result of model checking will be ''false/no'', i.e., the systematic model does not satisfy the required property.

In model checking, linear temporal logic [14], which was introduced to computer science by Turing Award winner Prof. Pnueli, and computational tree logic (CTL) [15], [16], which was proposed by Turing Award winner Prof. Clarke, are the two popular temporal logics. And these two logics have been used widely in international IT industry.

The state explosion problem is always one of the important bottlenecks of LTL model checking. To address this problem, many methods including symbolic, partial order reduction, equivalence, compositional reasoning, abstract and symmetry et al [1], have been proposed to reduce the huge state space, which is caused by the model checking algorithms. As a result, these methods work well. In a special case, 10^{120} states were verified automatically and effectively by a symbolic model checker [17]. However, the huge state space still restricts the further applications of model checking, in general situation.

Unfortunately, the state explosion problem inherently originates from the gene of model checking, LTL model checking in particular. Thus, no solution exists within the framework of hard computing (''actual/accurate computing'').

Motivated by it, we introduce machine learning to ''pretend'' to perform LTL model checking in this paper. In other words, we do not really execute MC computations, and we only ''predict'' results of LTL model checking using the machine learning algorithms. The key to design the new method is that both LTL model checking algorithms and ML binary classification ones will output and only output the following two opposite results after they runs: ''true''/''yes''/''1''/positive, or ''false''/''no''/''0''/negative. Thus, the LTL model checking is naturally induced to the ML binary classification. As a result, this famous non-polynomial complexity problem has an approximate polynomial solution. This is the contribution of this paper.

The remainder of this paper is organized as follows. In Section II, some preliminaries of model checking and machine learning are given. Section III presents our new method. The comprehensive experiments are conducted in section IV. Section V compares the new method with some related works. And Section VI will draw our conclusion.

II. PRELIMINARY

A. NuXMV/ NuSMV

NuSMV is developed by Carnegie Mellon University, University of Genova, University of Trento and FBK-IRST [18]. It is a free tool for symbolic CTL model checking and symbolic LTL model checking. As an improved version, NuXMV extends NuSMV, and the former features a strong verification engine based on state-of-the-art algorithms [19].

B. MACHINE LEARNING ALGORITHMS AND GRAPH LAB

One core goal of machine learning is to classify data. There are many ways for binary classification, such as BT, RF, DT and LR. These popular ML algorithms have been applied to many fields, such as text segmentation [20], face detection [21], hand pose recognition [22], multi-view, multi-pose object detection [23], emotion recognition [24], molecular hybridization prediction [25], computer vision [26], graphic and image [27], [28], Chess End Games [29], real-time packet classification [30], data stream mining [31], hard rock pillar stability prediction [32], classification of intra-subject MRI sequences [33] and controlling remote vehicles [34].

Graph Lab is an open source ML package [35], which was developed by Carnegie Mellon University. This tool integrates a variety of ML algorithms including BT, RF, DT and LR, which greatly simplifies the training process of models, and facilitates users' operations and implementation of specific ML algorithms.

III. THE PRINCIPLE OF THE NEW METHOD

We consider the following specific problem introduced in section I. How to determine whether systematic model K satisfies a LTL formula f or not, giving a pair of K and f using a ML algorithm.

The principle of our method is shown in Fig.1. The core is to train a large number of records using one machine learning algorithm. And each record contains information on a systematic models K, a LTL formulas f, and their model checking result r, i.e., flag. In other words, K and f are all the two ML features, while r is the only one ML label. Thus, a ML model called M which has a predictive ability is obtained.

The steps of the process can be described as follows.

–1). As shown in Fig.1 (a), one can run a LTL model checking algorithm and obtain a result of model checking for a given pair of K and f. On the basis of it, he or she can perform a binary classification. The result of the classification will be 1, if the result of model checking is ''true''. Otherwise, the result of the classification will be 0.

–2). Step 1) is repeated m_1 times, and a training set containing m_1 records is gotten. See the left part of Fig.1(b).

–3). Train this training set using a ML algorithm, and the ML model M is obtained. See the middle part of Fig.1 (b).

–4). Another pair of K' and f' whose model checking result is required to be predicted, is input to the trained model M. Whether K' satisfies f' or not? It can be predicted by M.

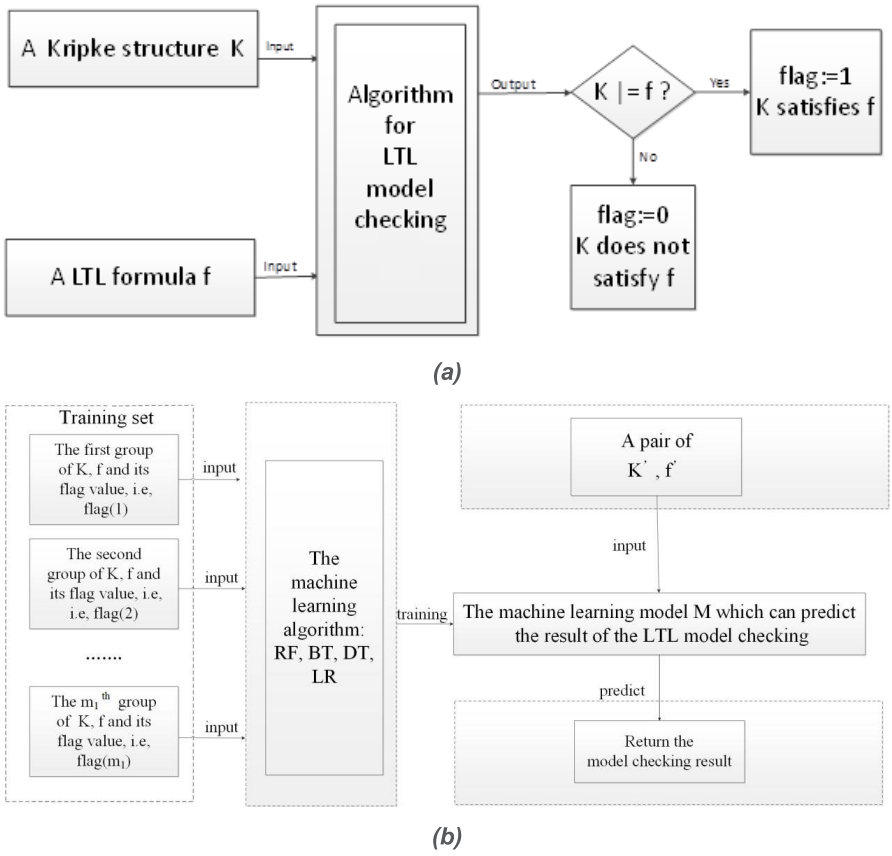


FIGURE 1. Given one pair of systematic model and formula, the new method can determine/predict whether this systematic model satisfies this formula or not (a) for a given pair of Kripke structure K and a LTL formula f, determine whether K satisfies f or not; (b) a ML model M which can predict the model checking result for a pair of K' and f', since M is obtained by training m1 groups of K, f and their model checking result.

IV. SIMULATED EXPERIMENTS

A. THE EXPERIMENTAL OBJECTIVE

We will explore the ability and the efficiency of the new method. Specifically, can the new method improve the efficiency significantly under the premise that the new method can approach the popular LTL model checking algorithm in terms of power?

B. THE SIMULATION PLATFORM

–1). CPU: Intel(R) Core(TM) i7-4770 CPU @3.40GHz.
–2). RAM: 16.0 G
–3). OS: Windows 7 64 bit.
–4). NuXMV and NuSMV: LTL model checking tool.
–5). Graph Lab: for implementing the several ML algorithms.
–6). Jupyter: a visual platform for some tools including Graph Lab and python.

C. EXPERIMENTAL PROCEDURES

–1). 200 LTL formulas f are generated randomly, where the length of each of formula equals to 500. In addition, 50 Kripke structures K are generated randomly. Thus, 200*50 = 10000 groups of sub-experiments on NuXMV will be conducted one by one for determining whether or not 50 Kripke structures satisfy 200 LTL formulas, respectively. In fact, we only select 100*50 = 5000 groups in all the 10000 groups, since other 5000 groups of model checking run very very slowly on NuXMV, even several days one group! In other words, 5000 samples are employed by the below ML experiments.

–2). We program on NuXMV for each pair of K and f one by one, and run our program so that the result of model checking is obtained.

–3). 5000 groups of sub-experiments on NuXMV for model checking produce 5000 records, where each record contains the three fields, for the two features and the one label, respectively. As a result, a data set containing 5000 records is obtained. And this is the original data set for our Graph Lab experiments.

–4). A part of 5000 records will take part in our training process on Graph Lab with a machine learning algorithm. These records form a training set, and other records form a test set. How many records are there in the training set? It depends on the value of some hyper-parameters. In fact, we only need adjust the values of the two hyper-parameters, i.e., seed and fraction, as presented in Table 2.

–5). We can obtain a ML model M, according to step 4). And we will get the predictive result in terms of model checking if we input the first two fields of a record in the test set, to M.

–6). We can compare the predicted result comes from step (5) with the real value of the third field, i.e., r. On the basis of it, we can make clear whether the prediction is accurate or not. On the basis of it, the average predictive accuracy of the whole test set can be computed. In this way, we can analysis the power of the new approach.

–7). We can obtain and compute the average running time for model checking one pair of K and f on NuXMV, as well as the average predictive time of one pair of K and f on Graph Lab, with the timing function in these two experimental tools. On the basis of it, we can compare the efficiency of the two methods.

From step 1) to step 7), 5000 groups sub-experiments on NuXMV and Graph Lab are conducted to study model checking long-length LTL formulas. In the similar way, sub-experiments on NuXMV and Graph Lab are conducted to study model checking short-length LTL formulas whose length is 25. In the similar way, NuSMV can replace NuXMV to do the above things.

In order to illustrate the principle and steps of our NuXMV experiments, we give an example as follows.

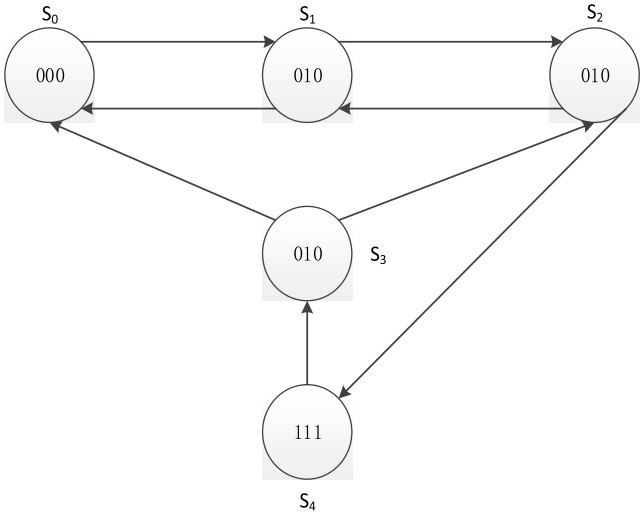


FIGURE 2. An example on Kripke structure K = K0.

Example 1 Fig.2 illustrates an example on Kripke structure K = K0. K0 has the five states and the eight transitions. All three atomic propositions p, q, r are not satisfied in state s_0, while only atomic proposition q is satisfied in state s_1, and so on. The state s_0 can be transformed to state s_1, and state s_1 can be transformed to state s_0 or state s_2, and so on. K0 can be represented with a string 00001001001011101101221243032 43. In this string, the first 15 bits describe whether the three atomic propositions are satisfied in each of the five states or not, while the rest bits represent the serial number of start state and the one of end state in all the eight transitions.

For K0 = "00001001001011101101221243032 43" and f1 = "!X((!F((!p&q|r)U(p|!q|r)))U(F(p&q&!r))) ", NuXMV model checking is run and returns it model checking result, i.e., "true (yes)", indicating K0 satisfy f1. Therefore, the three fields K, f, r of this record are "00001001001011101101221243032 43", "!X((!F((!p&q|r)U(p|!q|r)))U(F(p&q&!r)))" and "1", respectively. Moreover, NuXMV model checking spends 0.018 second.

As for K0 = "00001001001011101101221243032 43" and f2 = "X!((F(G!(!p|!q&r)))U((p&q|r)U(!p|q&r)))", its model checking result, i.e., "false (no)", is reported by NuXMV, which indicates K0 does not satisfy f2. Therefore, the three fields K, f, r of this record are "00001001001011101101221243032 43", "X!((F(G!(!p|!q&r)))U((p&q|r)U(!p|q&r)))" and "0", respectively. Moreover, NuXMV model checking spends 0.017 second this time.

Example 1 is over.

This example shows us how to construct two records with NuXMV and append them to our raw data set for our Graph lab experiments. It is just an example about short formulas, i.e., the length of the formulas is 25 (L = 25), due to simplicity. If the length of the formulas is 500 (L = 500), we will construct our raw data set in the same way.

Therefore, a data set A1 containing 405 records is obtained, where the length of each formula is 25, as well as a data set A2 containing 405 records is obtained where the length of each formula is 500. It should be noted that A1 and A2 use the same Kripke structures.

Furthermore, a data set A3 containing 5000 records is obtained where the length of each formula is 500, as well a data set A4 containing 1000 records is obtained where the length of each formula is 500. It should be noted that A3 and A4 are the two subsets of the data set originated from the above 10000 groups model checking. In fact, we have established another data set called A5 including 1000 records. Similar with A3 and A4, A5 has some randomly generated records and the length of each formula is 500. Differ from A3 and A4, A5 is not a sub-set of the data set originated from the above 10000 groups model checking.

A1, A2, A3, A4 and A5 provide the raw data for our Graph lab experiments.

D. EXPERIMENTAL RESULTS

1) THE EXPERIMENTAL RESULTS ON A1 AND A2

First, Fig.3 illustrates some comparisons of predictive accuracy and average predictive time for one record, when A1, A2 and the four ML algorithms are employed.

Supposing the classical model checking (such as NuXMV) has an accuracy of 1, the ML-based method has a max accuracy of 98.21% (L = 500), when LR algorithm is employed, as shown in Fig.3(a). Thus, the ML-based method has an approaching accuracy with state of the art of the classical model checking technique.

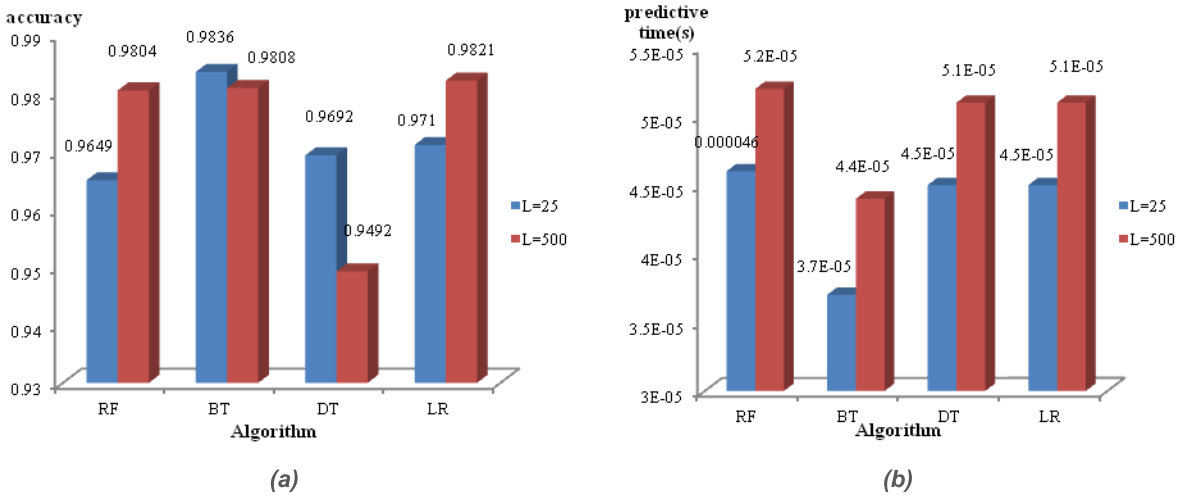


FIGURE 3. (chromatic figure) Power and efficiency of the new method under the circumstance of the different lengths of LTL formulas, i.e., L (A1 & A2 are used). (a) predictive accuracy; (b) average predictive time for one record.

TABLE 1. Compared with NuSMV and NuXMV, the new method enhances the efficiency of LTL model Checking (A1 and A2 are used).

Length of formulas, i.e., L	Average running time (t_1) of NuSMV for one pair of Kripke structure and formula	Average running time (t_2) of NuXMV for one pair of Kripke structure and formula	ML algorithms	Average predictive time (t_3) of the new method based on ML for one record (s)	t_1 / t_3	t_2/t_3
L=25	0.0150s	0.0150s	RF	0.000046	326	326
			BT	0.000037	405	405
			DT	0.000045	333	333
			LR	0.000045	333	333
L=500	227.28s	258.70s	RF	0.000052	4370769	4975000
			BT	0.000044	5165455	5879545
			DT	0.000051	4456471	5072549
			LR	0.000051	4456471	5072549

As shown in table 1, each of the four ML algorithms obtains their predictive results on one group model checking within 0.00006 seconds, when the length of LTL formulas is 500. In comparison, NuXMV consumes 258.7 seconds for one group model checking. Compared with NuXMV model checking, the efficiency of the ML-based method increases several million times, when the length of LTL formulas is 500, as shown in table 1. And the efficiency of the ML-based method increases only several hundred times, when the length of LTL formulas is 25. This discovery indicates that the longer the formula, the more comparative efficiency the ML-based method has.

Thus, we will study what happen when the length of LTL formulas is 500 rather than 25. A data set containing 405 records is too small. Thus, A3 will be employed.

2) THE EXPERIMENTAL RESULTS ON A3

Now, the data set has 5000 records. And the results are illustrated in table 2 and table 3.

This time, as illustrated in table 2, the max accuracy reaches 1 when LR is used, indicating that the ML-based method has a same power with state of the art of the classical model checking technique, although the former method is an approximate model checking one and the latter method is based on accurate computing.

In addition, the ML-based method is several million times faster than state of the art of the classical model checking technique again, as shown in table 3.

E. DISCUSSION

First, 1160 groups of model checking results are "yes", and other 3840 groups of results are "no", for 5000 groups of model checking experiments in the data set A3. These results provide abundant positive examples and negative ones, avoiding the data unbalance, which can guarantee the generalization ability of ML model.

Second, as shown in table 2 (1), the max predictive accuracy of the machine learning algorithms is 1 when the length of the formula is 500. It indicates that the predictions are very accurate using the new method to simulate LTL model checking, regardless of the length of the LTL formulas. The reason for this is that the LTL model checking is a strongly learnable problem. Therefore, the new method based on machine learning has a good learning ability.

It should be noted that, this max accuracy of 100% is obtained with 492 test records in our experiments, so that

TABLE 2. Graph Lab experiments where length of each formula is 500 (A3 is used): (1) the optimal results (AUC: Area under ROC Curve; TNR: Prediction specificity; TPR: Prediction sensitivity); (2) What are the values of the parameters and the hyper-parameters if the illustrations of Table 2(1) occur (Meaning of parameters & hyper-parameters: Seed: Seed for the random number generator used to split; Fraction: For determining the proportion of the records of training set in the total records of data set; Testing record #: The number of test records).

(1)

Algorithms	RF	BT	DT	LR
Prediction Accuracy	0.8918	0.9052	0.8560	1
Running time per record (in second)	0.000049	0.000036	0.000033	0.000032
AUC	0.7244	0.9792	0.6138	1
Specificity (TNR)	0.996	0.991	0.994	1
Sensitivity (TPR)	0.454	0.546	0.234	1
Precision	0.964	0.937	0.897	1
Prediction Accuracy	0.8918	0.9052	0.8560	1
Running time per record (in second)	0.000049	0.000036	0.000033	0.000032

(2)

Algorithms	RF	BT	DT	LR
Training record #	4381	4441	4389	4508
Testing record #	619	559	611	492
Seed	2008	2280	1750	135
Fraction	0.88	0.89	0.88	0.9

TABLE 3. Compared with NuSMV and NuXMV, the new method enhances the efficiency of LTL model Checking (L = 500 & A3 is used).

Average running time (t_1) of NuSMV for one pair of Kripke structure and formula	Average running time (t_2) of NuXMV for one pair of Kripke structure and formula	ML algorithms	Average predictive time (t_3) of the new method based on ML for one record (s)	t_1 / t_3	t_2/t_3
208.3s	201.5s	RF	0.000049s	4251020	4112245
		BT	0.000036s	5786111	5597222
		DT	0.000033s	6312121	6106061
		LR	0.000032s	6509375	6296875

the actual max accuracy p_{max} is not less than $492/(492 + 1) = 492/493 = 99.8\%$. In other words, $p_{max} \in [0.998, 1]$ holds.

As shown in table 3, the average running time of each of machine learning algorithms for predicting one record is less than 0.0001 seconds, when the length of the formula is 500. It indicates that the predictions are very fast using the new method to simulate LTL model checking, although the formulas are very long. The reason for this is that the LTL model checking is a strongly learnable problem, so that it can be simulated in polynomial times. By contrast, the average running time of state of art of the algorithm of the classical LTL model checking technique is more than 200 seconds when the length of the formula is 500, which is more than 4 million times as much as the new method. The reason is that the LTL model checking algorithm has a non-polynomial complexity, while the new method based on machine learning has a polynomial complexity.

Third, as shown in Table 1, compared with the LTL model checking algorithm, the new method will enhance the efficiency 300 times at least if the length of all formulas is 25, whereas the new method will enhance the efficiency 4 million times at least if the length of all formulas is 500. This phenomenon prompts us that the longer the length of the formula, the higher the efficiency of the new method is, due to the advantage of the polynomial algorithm over the non-polynomial algorithm.

It should be noted that the LTL model checking problem has an inherent exponential complexity, and all LTL model checking algorithm can never break this low bound. In comparison, as an approximate model checking method rather than accurate model checking one, the new method based on machine learning algorithm breaks this limitation.

Four, as shown in table 2 (1), the optimal accuracies of the different ML algorithms for predicting the results of LTL model checking are different. The accuracy of LR is highest, i.e., 100%, and the accuracies of the other three algorithms are lower than 91%. Considering model checking is a kind of accurate computing, it is safe to say that BT, RF and DT is not suitable to this mission, whereas LR is better.

Final, the thing is, the trained ML model can predict the model checking result for a given arbitrary Kripke structure and a given arbitrary LTL specification, if the Kripke structures in the testing set and the training set have the same size, and the LTL formulas in the testing set and the training set have the same length. What is to be done if the length and the size are different? The answer is simple. One can train the different ML models for predicting the different class of formulas and Kripke structures. The formulas which have the same length and the Kripke structures which have the

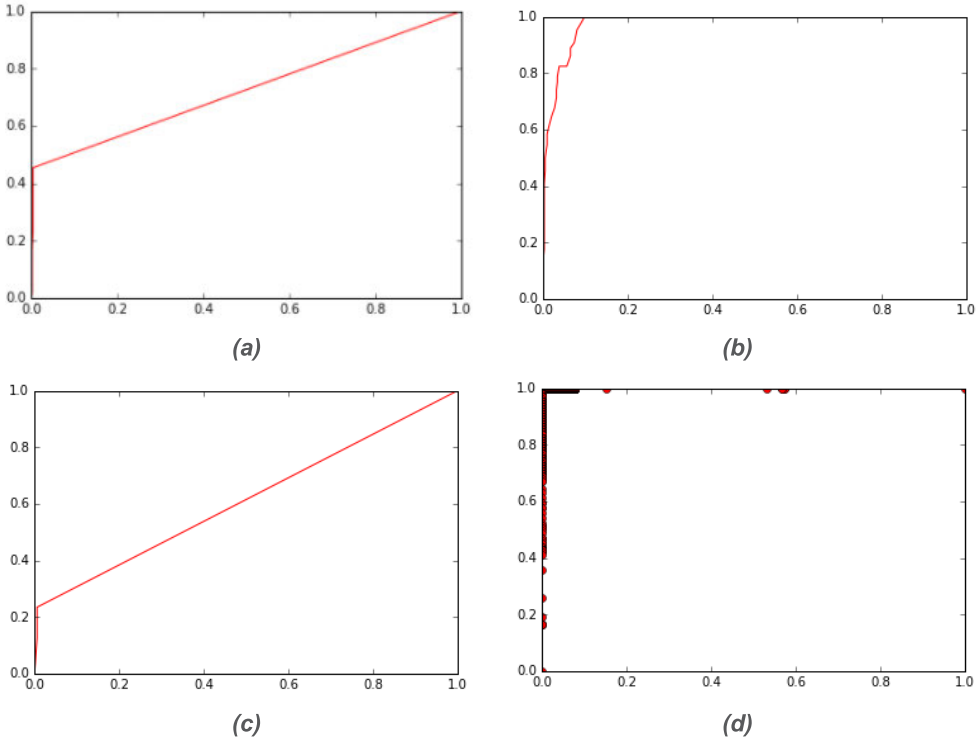


FIGURE 4. (chromatic figure) Roc Curve of the Optimal Classifiers, (L = 500 & A3 is used). (a) RF; (b) BT; (c) DT; (d) LR.

same size will be put into the same class. Lengths and sizes, i.e., the number of classes, are finite and enumerable, so that the number of ML models is finite and enumerable. If someone conducts model checking using the new method, it will automatically select an appropriate ML model according to lengths and sizes of input formula and Kripke structure.

F. MORE COMPARISONS AMONG THE DIFFERENT ML ALGORITHMS

How to evaluate the obtained the different ML models mentioned above? Besides predictive accuracy, the researchers often use AUC, ROC curve, sensitivity, specificity and precision, as the performance metrics of ML binary classifiers.

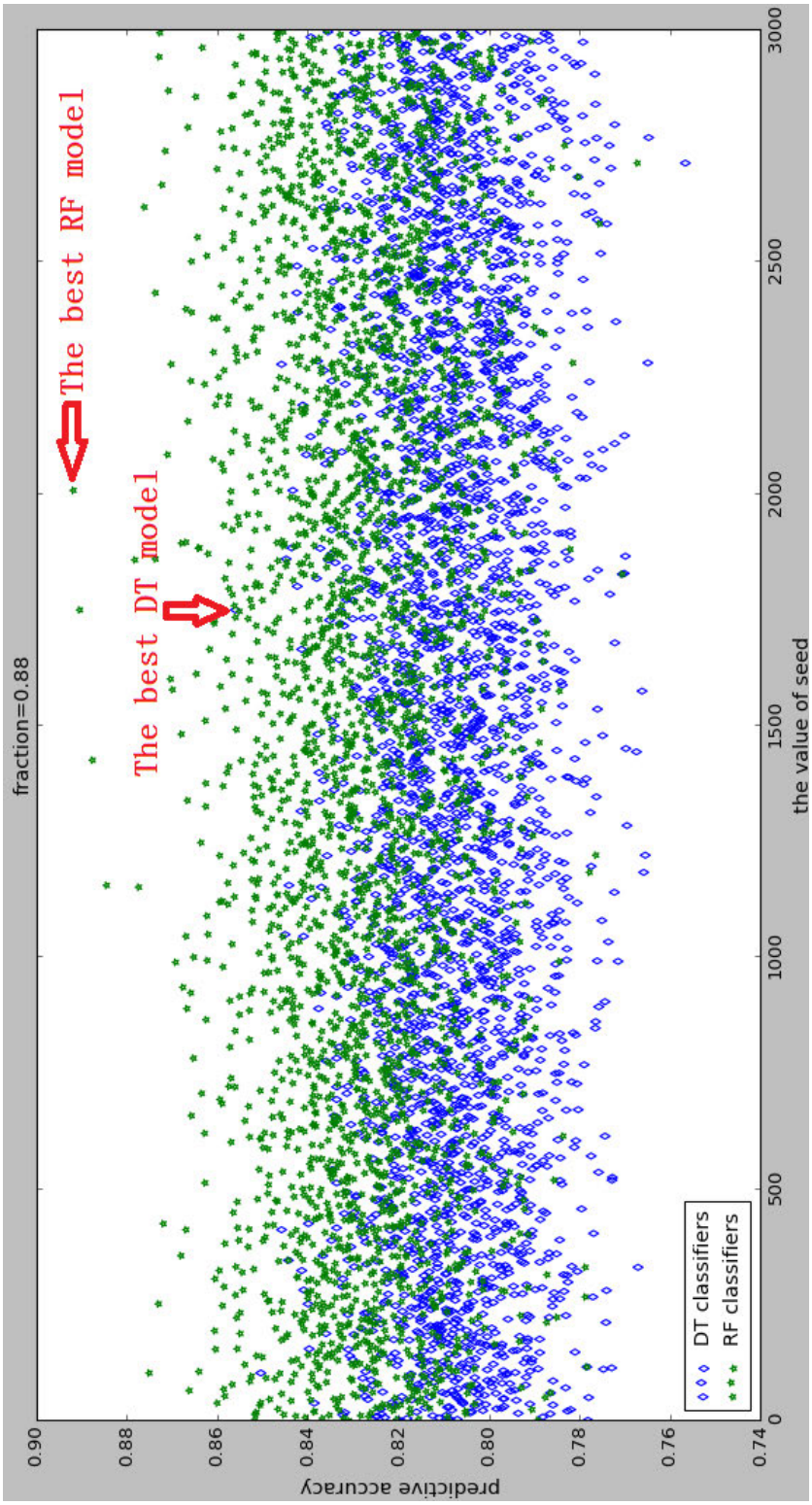
Sensitivity (also called TPR) and specificity (also called TNR) are statistical measures of the performance of a binary classification test, where sensitivity measures the proportion of actual positives that are correctly identified as such, and specificity measures the proportion of actual negatives that are correctly identified as such. In addition, precision measures the proportion of true positives to predicted positives. Table 2(1) shows sensitivity, specificity and precision for the four optimal classifiers of the four ML algorithms, when the length of the LTL formulas is 500.

ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier since its discrimination threshold is varied. In short, the closer ROC curve is to the top left corner of the graph, the better the classifier is. AUC is defined as the area under the ROC curve. Obviously, the value of this area will not be greater than 1. AUC ranges from 0.5 to 1 because ROC curves are generally above the line y = x. AUC is the most important evaluation criteria for binary classifiers because it's not easy for users to distinguish which ROC curve is closer to the top left corner of the graph, i.e., which classifier works better. In comparison, AUC can do it since this is a numerical value. That is to say, the bigger the value of AUC is, the better the classifier is.

Fig.4 illustrates the ROC curves for all the four optimal classifiers, as well as table 2(1) shows the values of AUC for each of the four optimal classifiers, when the length of the LTL formulas is 500. Plainly, this discovery prompts us that BT and LR are the suitable algorithms to our application, and the best classifiers can occur if each of these two algorithms is employed, when the length of the LTL formulas is 500. However, LR is better than BT, as shown in this table.

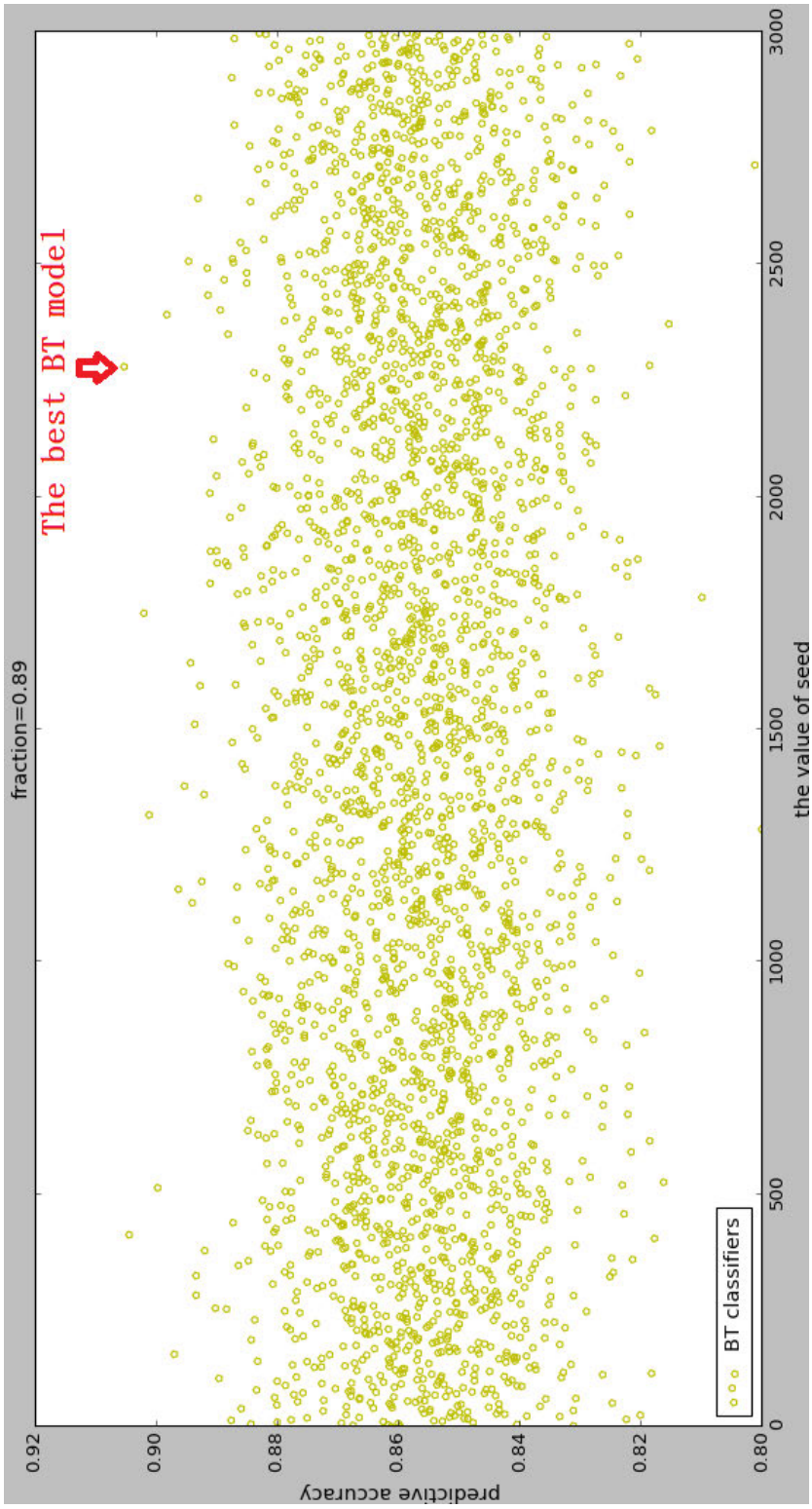
In the above discussion, our program running on Graph Lab automatically seeks the highest predictive accuracy for each of the four ML algorithms, respectively, by adjusting the values of the two hyper-parameters including the one that can affect the number of records in the test set. As a result, the corresponding four test sets are obtained automatically, and they contain different number of records, when the four optimal accuracies of the four ML algorithms appear. That is to say, it is the different values of the hyper-parameters that lead to the optimal models of the different algorithms.

Now, the obvious question becomes: for the four ML algorithms: how will the observation be if the four ML algorithms are set to the identical values of the hyper-parameters? Fig.5 has answered this question.



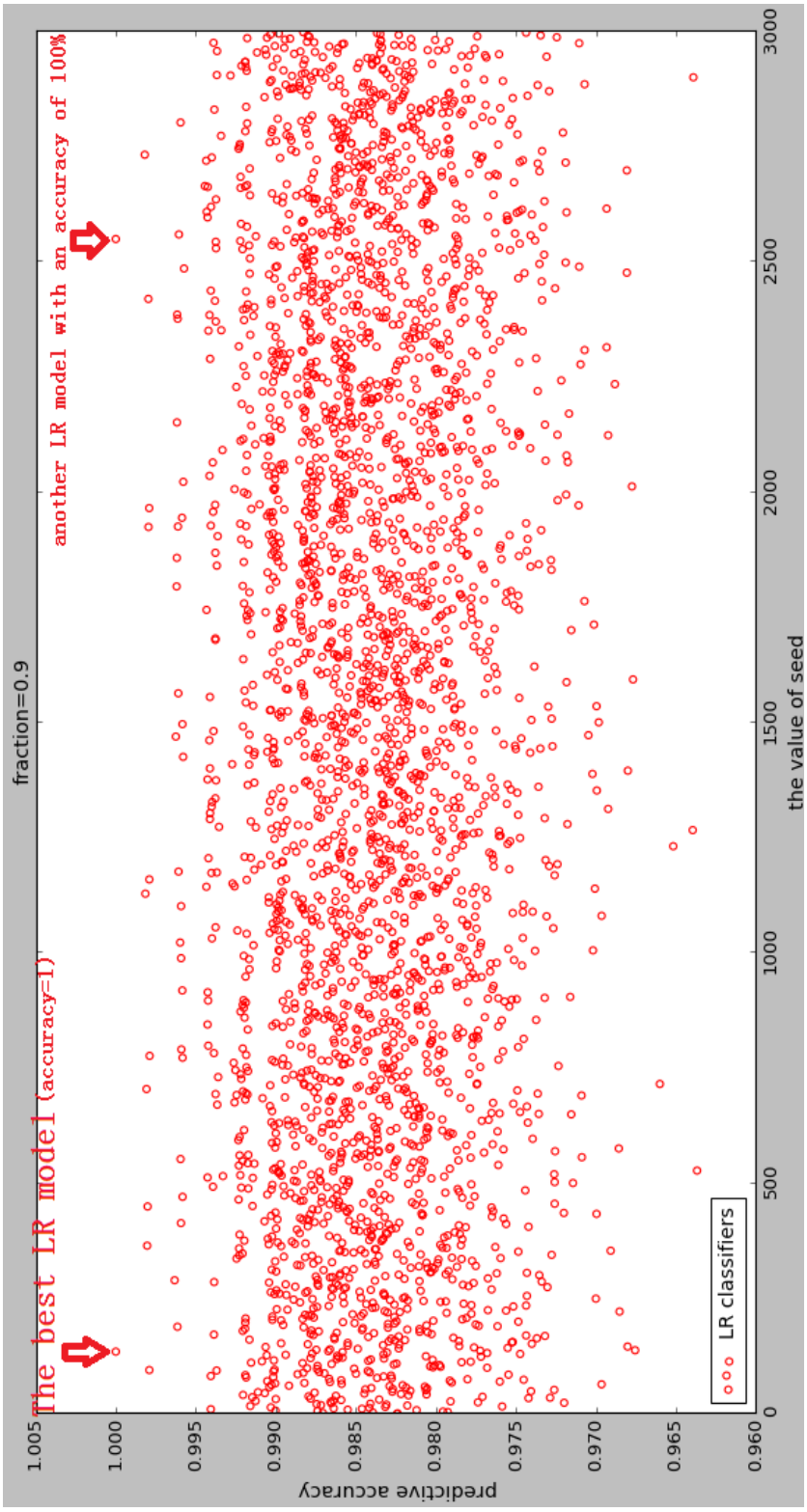
(a)

FIGURE 5. (chromatic figure) Comparison of performance of ML algorithms with the same values of hyper-parameters (seed & fraction) (L = 500 & A3 is used). (each dot denotes a classifier, its abscissa means the value of seed, and its ordinate means the obtained accuracy of the classifier). (a) obtained various classifiers when fraction = 0.88; (b) obtained various classifiers when fraction = 0.89; (c) obtained various classifiers when fraction = 0.9; (d) obtained various classifiers (a combination of the above three sub-figures).



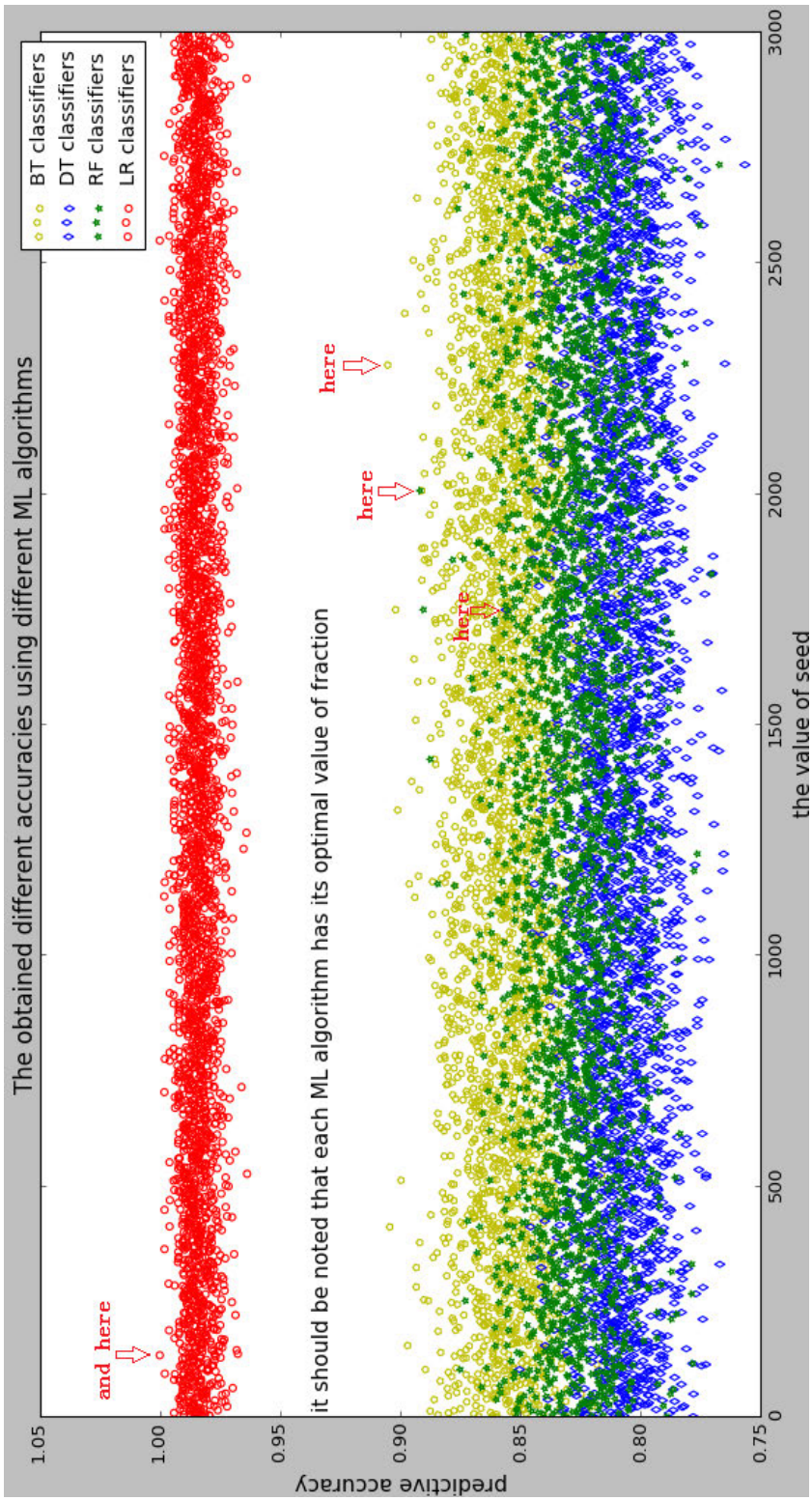
(b)

FIGURE 5. *(continued.)* **(chromatic figure) Comparison of performance of ML algorithms with the same values of hyper-parameters (seed & fraction) (L = 500 & A3 is used). (each dot denotes a classifier, its abscissa means the value of seed, and its ordinate means the obtained accuracy of the classifier). (a) obtained various classifiers when fraction = 0.88; (b) obtained various classifiers when fraction = 0.89; (c) obtained various classifiers when fraction = 0.9; (d) obtained various classifiers (a combination of the above three sub-figures).**



(c)

FIGURE 5. *(continue.)* **(chromatic figure) Comparison of performance of ML algorithms with the same values of hyper-parameters (seed & fraction) (L = 500 & A3 is used). (each dot denotes a classifier, its abscissa means the value of seed, and its ordinate means the obtained accuracy of the classifier). (a) obtained various classifiers when fraction = 0.88; (b) obtained various classifiers when fraction = 0.89; (c) obtained various classifiers when fraction = 0.9; (d) obtained various classifiers (a combination of the above three sub-figures).**



(d)

FIGURE 5. *(continue.)* **(chromatic figure) Comparison of performance of ML algorithms with the same values of hyper-parameters (seed & fraction) (L = 500 & A3 is used). (each dot denotes a classifier, its abscissa means the value of seed, and its ordinate means the obtained accuracy of the classifier). (a) obtained various classifiers when fraction = 0.88; (b) obtained various classifiers when fraction = 0.89; (c) obtained various classifiers when fraction = 0.9; (d) obtained various classifiers (a combination of the above three sub-figures).**

TABLE 4. Graph Lab experiments where length of each formula is 500 (A4 is used).

Algorithms	RF	BT	DT	LR
Prediction Accuracy	0.9590	0.9970	0.9360	1
Running time per record (in second)	0.000035	0.00005	0.000038	0.000035
AUC	0.9970	0.9983	0.8040	1
Specificity (TNR)	0.995	0.996	0.996	1
Sensitivity (TPR)	0.764	1	0.611	1
Precision	0.968	0.981	0.970	1

TABLE 5. Compared with NuSMV and NuXMV, the new method enhances the efficiency of LTL model Checking (L = 500 & A4 is used).

Average running time (t_1) of NuSMV for one pair of Kripke structure and formula	Average running time (t_2) of NuXMV for one pair of Kripke structure and formula	ML algorithms	Average predictive time (t_3) of the new method based on ML for one record (s)	t_1 / t_3	t_2/t_3
208.3s	201.5s	RF	0.000035s	5951429	5757143
		BT	0.00005s	4166000	4030000
		DT	0.000038s	5481579	5302632
		LR	0.000035s	5951429	5757143

This figure illustrates the relationship between all the two hyper-parameters and the obtained accuracies. In this figure, the different colored dots which have the same value of the horizontal ordinate denote the different performance of the different ML algorithms with the same value of the hyper-parameters.

As shown in Fig.5, we evidently feel that the red dots are a little higher than the other dots on the whole. It indicates that the overall performance of LR is better than those of the other three algorithms, when the length of the LTL formulas is 500.

Furthermore, each ML algorithm has a best classifier, i.e., ML model, as shown in the red arrows of the figure. Clearly, the location of the best LR classifier is higher than those of the other three best classifiers, according to the location of the four red arrows. It also indicates that the optimal accuracy of LR is better than those of the other three ML algorithms.

G. MORE COMPARISONS ON THE DIFFERENT DATA

In the above discussion, we give some comparisons among the four ML algorithms on the same raw data set rather than the same test set. In other words, given a fixed data set, which an algorithm will perform well, if both training and testing use this data set? The above experimental results demonstrate that LR is the best.

Now, another obvious question comes: how to compare these four ML algorithms on the same test set instead of the same raw data set? In other words, given a fixed test set, which an algorithm and its optimal classifier will perform well, if the four optimal classifiers use this same test set? The experiments on A4 will provide an answer.

This time, all the four optimal classifiers run on the same test set A4. As shown in table 4 and table 5, LR and BT provide the two fast and accurate classifiers, where the LR optimal model is best again. Their AUC values identify the same viewpoint, as illustrated in Fig.6. As for the average predictive time for one record, all the four optimal classifiers are very fast again, as shown in table 5.

TABLE 6. Graph Lab experiments where length of each formula is 500 (A5 is used).

Algorithms	RF	BT	DT	LR
Prediction Accuracy	0.3	0.3	0.3	0.3

Now, let A4 be replaced by A5, and all the four optimal classifiers run on the same test set A5. The results are depicted as table 6, indicating the performance is not acceptable at all! In fact, the combination of table 4 and table 6 shows a vital difference with regard to the following precondition of the new approach.

Let $A_L = \{l_1, l_2, \ldots, l_n\}$ and $A_K = \{k_1, k_2, \ldots, k_m\}$, thus the set $A_L \times A_K$ has m*n elements. If each element of A_L is a LTL formula and each element of A_K is a Kripke structure, each element of $A_L \times A_K$ will be a model checking result. If a subset of $A_L \times A_K$ makes up a training set and another subset of $A_L \times A_K$ forms a test set, the optimal classifiers will perform well. However, even the optimal classifiers are hard to complete their mission successfully, if a testing sample does not occur in any subset of $A_L \times A_K$. In our experiments, 10000 records mentioned in section IV.C make up $A_L \times A_K$, and the subsets of $A_L \times A_K$ are A3 and A4, not A5. It is this fact and reason that leads to the sharp difference between table 4 and table 6.

In summarize, the new method is much faster than the existing ones, as well as the former is almost equal to the latter in terms of accuracy. However, the use of the new method would be limited by the above constraint, if a big data set could not be established in advance.

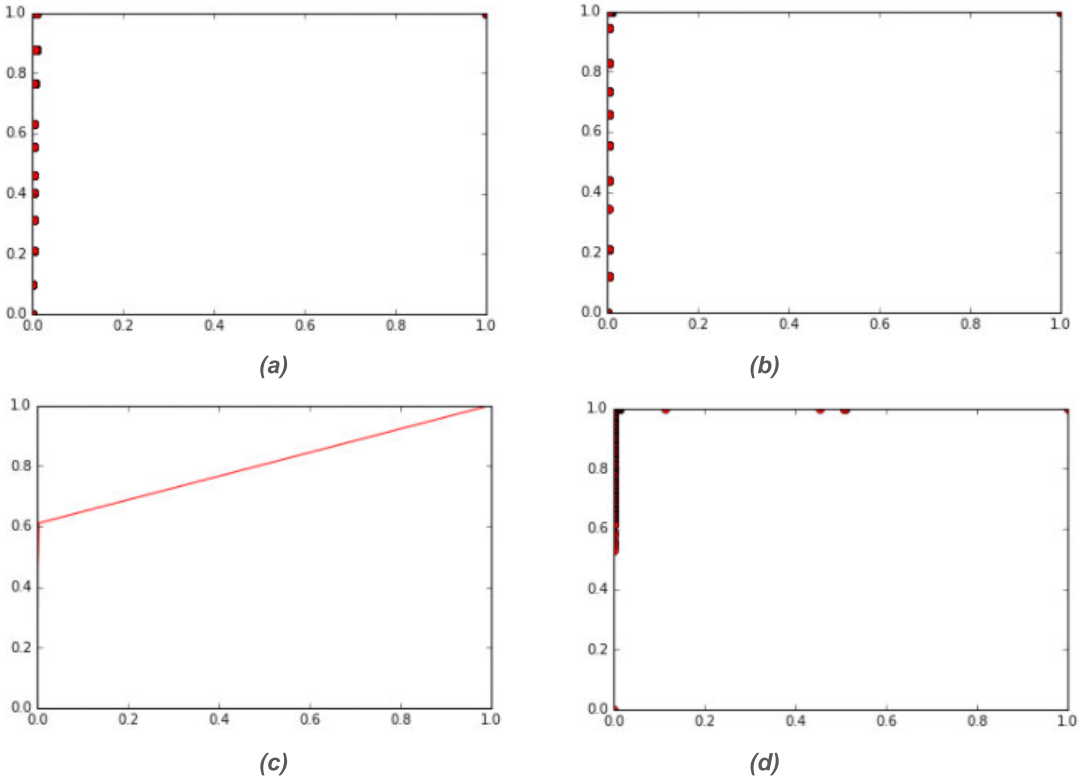


FIGURE 6. (chromatic figure) Roc Curve of the Optimal Classifiers, (L = 500 & A4 is used). (a) RF; (b) BT; (c) DT; (d) LR.

TABLE 7. The key differences among some related works and the new method.

	The distinguishing feature of this type of studies	The distinguishing feature of the new method
Studies Type 1)	deal with some SAT problems rather than MC ones	address the key MC problem directly
Studies Type 2)	ML and MC cooperate for a third party	perform ML for MC
Studies Type 3) & Studies Type 4)	The employed ML does NOT change the trunk of MC core engine, which is still based on state space exploration, so that both the state explosion problem and non-polynomial complexity can NEVER be avoided fundamentally	The employed ML has changed the trunk of MC core engine, and the new method is based on data training & prediction rather than state space exploration, so that both the state explosion problem and non-polynomial complexity are avoided

TABLE 8. Comparison among the existing MC algorithms of and the new method.

	The existing MC algorithms	The new method
State explosion problem	Can be relieved instead of avoided	Has been avoided
Non-polynomial time complexity	They are suffering from it	Has been avoided
Before real customers use it	A model checker is need to be developed	Not only a ML-based tool, but also a massive data set is needed. It may be quite an enormous undertaking
Applied range	Very wide	Has some limitations, such as safety-critical systems
Can counterexamples be generated?	Yes	No

V. THE RELATED WORKS

A. SOME STUDIES RELATED TO BOTH ML AND MC

Model checking is a mainstream technique in the theoretical computer science community, while machine learning is a mainstream technique in the artificial intelligence community. Up to now, a number of studies are related with both of these two mainstream techniques. These studies are very important and they can be summarized as the following three types, roughly speaking:

–1). Machine learning for some SAT solvers [36]–[38]. In the strict sense, these works relate to machine learning, not model checking. In fact, SAT denotes the

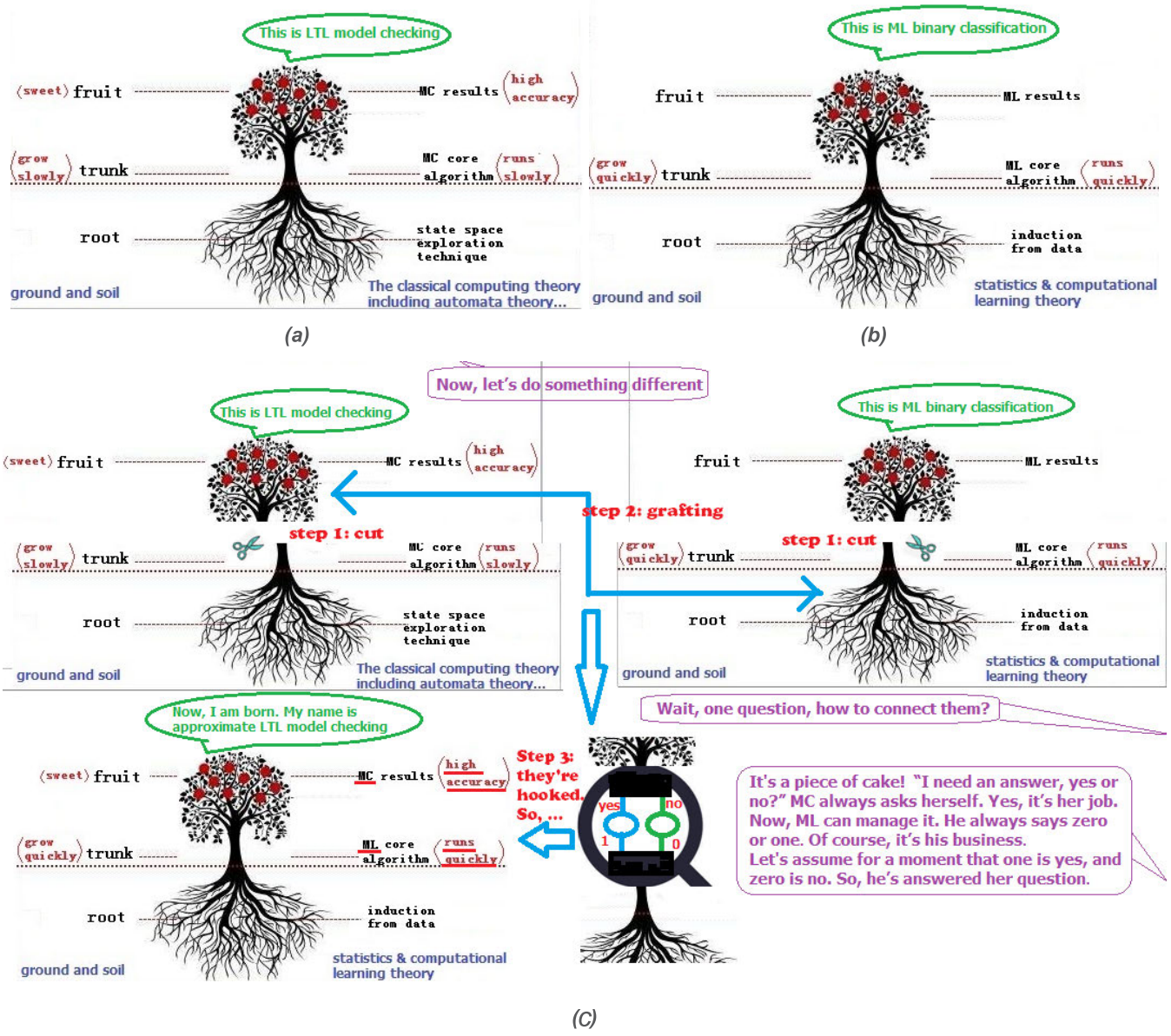


FIGURE 7. (a chromatic figure) Illustrations of something about our study. (a) about LTL model checking; (b) about ML binary classification; (c) an illustration of this study.

boolean satisfiability. Therefore, SAT solvers deal with boolean formulas, whereas model checking deal with temporal logic formulas. And satisfiability and model checking are relevant and different.

–2). These studies employ both model checking and machine learning to address some problems in the field of a third party. These works are quite far from this paper in terms of topics. Thus, we will not dwell on it here.

–3). In fact, some studies have employed machine leaning to deal with some model checking topics. In such researches including the new method in this paper, machine learning provides technical services, and model checking is a user of machine learning. For examples, how to select a suitable software model checker from a number of software model checking tools for a given input instance? To this end, an approach called Mux was presented to perform machine learning on a repository of software verification instances [42]. Another study is about on-the-fly model checking. Since in on-the-fly model checking, one do not have complete knowledge about the model, the authors of Ref.[43] use a machine learning method based on interaction and reward receiving. More studies, such as [44], [45], employ machine learning to deal with some topics related with refining the abstraction, a technical scheme that can reduce state space in the process of model checking. It is a hotspot to study statistical model checking, a variant technique of model checking, using machine learning. See

Ref. [46]–[49], [50], [51] and [52] for more details. And no more information will be detailed here, due to the limitation of space.

Table 7 depicts the essential differences among the new method and some related works.

B. COMPARISON WITH THE EXISTING LTL MODEL CHECKING

Due to the characteristics of machine learning, it is not guaranteed that a ML-based method can steadily reach the predictive accuracy of 100% in any case. Thus, it is not recommended to use the newly proposed approximate LTL model checking technique to safety-critical systems. In comparison, the field of safety-critical systems has been one of the most important applications of the classical LTL model checking.

In addition, counterexamples cannot be produced since no path or state is generated in the process of running the ML-based method. Thus, it is not recommended to use the new method to the MC cases that need counterexamples.

Furthermore, as mentioned above, pre-construction of a massive data set will be a very time-consuming mission in the stage of development. Maybe online learning can help engineers deal with this problem. Anyway, it is just a cost in the stage of realization in engineering.

Obviously, the new method has its flaws. However, the new one also shows its comparative advantages. All the existing model checking methods are based on accurate computing, and the core of them is to explore exhaustively state space. By contrast, our method based on machine learning is no longer an accurate computing, although the result is still accurate. The new method never searches the state space at all, and it just does some predictions according to a data set. As a result, the state explosion problem and the exponential complexity are avoided entirely.

According to the experimental results and analysis, the existing LTL model checking approaches and the new one have the different characteristics, as summarized in table 8. Thus, it is safe to say that they complement each other.

C. COMPARISON WITH SOME WORKS NAMING APPROXIMATE MODEL CHECKING

Some studies have been conducted with naming ''approximate model checking''. To the best of our knowledge, there are the following three situations, roughly speaking:

–1). In some MC extensions, such as stochastic model checking or probabilistic model checking, the complexity and the state explode are more serious than that of the model checking without extension. Even some time complexities reach non-elementary, beyond exponent! If systematic models [53], [54] and formulas [55], [56] are dealt with approximately, or paths from the state space are obtained through random sampling [57], such approximate technique can reduce the state space or complexity [58], [59].

–2). Hybrid systems are often undecidable so that they cannot be conducted model checking. If some techniques including approximate ones [60] are employed to simplify the decidability, model checking will be available.

–3). In order to reduce the state space, the following mean are presented. For the formula describing a property, an upper bound formula and a lower bound formula are provided, so that the logically the lower bound formula implies the original formula, as well as the original formula implies the upper bound formula. That is, if the lower bound formula is verified as true, then the original formula must also hold in the given model. Conversely, if the upper bound formula is evaluated to false, then the original formula must also be false [61]. If the upper bound formula and a lower bound formula are easier to be conducted model checking, the the state space will be reduced. This idea was pioneered in Ref. [62], which has a similar methodology with [63] and [64].

Obviously, the meaning of the term ''approximate model checking'' is changing and confusing. One should judge its true meaning according to context. Recording these studies mentioned in this subsection as type 4), the last line in table 7 formulates the fundamental differences between this kind of studies and the new method. In a word, these works are very important, but they are NOT the same kind of things with the new method at all.

VI. CONCLUSION

In this paper, an approximate LTL model checking technique is formed and pioneered. To the best of our knowledge, this is the first approach that uses directly machine learning as the core engine of model checking, as well as the first model checking method that avoid state explosion problem totally. Fig.7 illustrates this study.

Our experiments have demonstrated that the longer the LTL formula, the more obvious the comparative advantage of the new methods is. In the real world, state of the art of LTL model checking is insufficient to verify large-scale operating systems and their complex properties, due to the well-known state explosion problem and the exponential complexity. This background will help us understand the benefit of using the new method.

REFERENCES

[1] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.

[2] J. Barnat, P. Bauch, L. Brim, and M. Češka, ''Designing fast LTL model checking algorithms for many-core GPUs,'' *J. Parallel Distrib. Comput.*, vol. 72, no. 9, pp. 1083–1097, 2012.

[3] R. Carbone, ''LTL model-checking for security protocols,'' *AI Commun.*, vol. 24, no. 3, pp. 281–283, 2011.

[4] F. Song and T. Touili, ''Model-checking for Android malware detection,'' in *Programming Languages and Systems*. Cham, Switzerland: Springer, 2014.

[5] E. Clarke. (2005). *Grand Challenge Problem: Model Check Concurrent Software*. [Online]. Available: http://www.csl.sri.com/users/shankar/VGC05/Clarke.ppt

[6] *NASA Systems Engineering Handbook, Revision 1*, document NASA/SP-2007-6105, NASA, Washington, DC, USA, 2007.

[7] N. Vittoria, A. Santone, M. Tipaldi, and L. Glielmo, ''Probabilistic model checking applied to autonomous spacecraft reconfiguration,'' in *Proc. IEEE Int. Workshop Metrol. Aerosp., MetroAeroSpace*, Jun. 2016, pp. 556–560. doi: 10.1109/MetroAeroSpace.2016.7573276.

[8] H. T. T. Doan, F. Bonnet, and K. Ogata, "Model checking of robot gathering," in *Proc. 21st Int. Conf. Principles Distrib. Syst. (OPODIS)*, 2017, pp. 1–16. doi: 10.4230/LIPIcs.OPODIS.2017.12.
[9] M. Brambilla, A. Brutschy, M. Dorigo, and M. Birattari, "Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, no. 4, Jan. 2015, Art. no. 17.
[10] M. L. Bolton and E. J. Bass, "Generating erroneous human behavior from strategic knowledge in task models and evaluating its impact on system safety with model checking," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 6, pp. 1314–1327, Nov. 2013.
[11] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Using formal verification to evaluate human-automation interaction: A review," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 3, pp. 488–503, May 2013.
[12] C. Wang, L. Cao, and C.-H. Chi, "Formalization and verification of group behavior interactions," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 8, pp. 1109–1124, Aug. 2015.
[13] *Microsoft Research—Emerging Technology, Computer, and Software Research*. Accessed: Sep. 21, 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/cmc-a-pragmatic-approach-to-model-checking-real-code/
[14] A. Pnueli, *The Temporal Logic of Programs*. Rehovot, Israel: Weizmann Science Press of Israel, 1977.
[15] M. Ben-Ari, A. Pnueli, and Z. Manna, "The temporal logic of branching time," *Acta Inform.*, vol. 20, no. 3, pp. 207–226, 1983.
[16] E. A. Emerson and E. M. Clarke, "Using branching time temporal logic to synthesize synchronization skeletons," *Sci. Comput. Program.*, vol. 2, no. 3, pp. 241–266, 1982.
[17] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill, "Symbolic model checking for sequential circuit verification," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 4, pp. 401–424, Apr. 1994.
[18] FBK. (2016). *NuSMV: A New Symbolic Model Checker*. [Online]. Available: http://nusmv.fbk.eu/
[19] FBK. (2018). *The nuXmv Model Checker*. [Online]. Available: https://es-static.fbk.eu/tools/nuxmv/
[20] X. Peng, S. Setlur, V. Govindaraju, and S. Ramachandrula, "Using a boosted tree classifier for text segmentation in hand-annotated documents," *Pattern Recognit. Lett.*, vol. 33, no. 7, pp. 943–950, 2012.
[21] C. Demirkir and B. Sankur, "Face detection using boosted tree classifier stages," in *Proc. IEEE 12th Signal Process. Commun. Appl. Conf.*, Apr. 2004, pp. 575–578.
[22] T. Parag and A. Elgammal, "Unsupervised learning of boosted tree classifier using graph cuts for hand pose recognition," in *Proc. Brit. Mach. Vis. Conf.*, Edinburgh, U.K., Sep. 2013, pp. 1259–1268.
[23] B. Wu and R. Nevatia, "Cluster boosted tree classifier for multi-view, multi-pose object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.
[24] M. Day, "Emotion recognition with boosted tree classifiers," in *Proc. 15th ACM Int. Conf. Multimodal Interact.*, 2013, pp. 531–534.
[25] W. Zhu, Y. Han, H. Wu, Y. Liu, X. Nan, and Q. Zhou, "Predicting the results of molecular specific hybridization using boosted tree algorithm," *Concurrency Comput., Pract. Exper.*, to be published. doi: 10.1002/cpe.4982.
[26] G. Yu, J. Yuan, and Z. Liu, "Unsupervised random forest indexing for fast action search," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 865–872.
[27] C. Shen, Y. Chen, G. Yang, and X. Guan, "Toward hand-dominated activity recognition systems with wristband-interaction behavior analysis," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2018.2819026.
[28] A. Bera and D. Bhattacharjee, "Human identification using selected features from finger geometric profiles," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2017.2744669.
[29] J. R. Quinlan, "Learning efficient classification procedures and their application to chess end games," *Mach. Learn.* Berlin, Germany: Springer, 1983, pp. 462–482.
[30] X. Dong, M. Qian, and R. Jiang, "Packet classification based on the decision tree with information entropy," *J. Supercomput.*, vol. 74, pp. 1–15, Jan. 2018.
[31] M. Jaworski, P. Duda, and L. Rutkowski, "New splitting criteria for decision trees in stationary data streams," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2516–2529, Jun. 2018.
[32] J. Zhou, X. Li, and H. S. Mitri, "Comparative performance of six supervised learning methods for the development of models of hard rock pillar stability prediction," *Natural Hazards*, vol. 79, no. 1, pp. 291–316, 2015.
[33] Y. Zhang, D. Kwon, and K. M. Pohl, "Computing group cardinality constraint solutions for logistic regression problems," *Med. Image Anal.*, vol. 35, pp. 58–69, Jan. 2017.
[34] M. McClelland and M. Campbell, "Probabilistic modeling of anticipation in human controllers," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 4, pp. 886–900, Jul. 2013.
[35] Turi. (2018). *Graphlab Create. Fast, Scalable Machine Learning Modeling in Python*. [Online]. Available: https://turi.com/
[36] S. Haim and T. Walsh, "Restart strategy selection using machine learning techniques," in *Proc. 12th Int. Conf. Theory Appl. Satisfiability Test.* Berlin, Germany: Springer, 2009.
[37] J. H. Liang, C. Oh, M. Mathew, C. Thomas, C. Li, and V. Ganesh, "Machine learning-based restart policy for CDCL SAT solvers," in *Theory and Applications of Satisfiability Testing—SAT*. Oxford, U.K.: Springer, 2018.
[38] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, "Learning rate based branching heuristic for SAT solvers," in *Theory and Applications of Satisfiability Testing—SAT*. Cham, Switzerland: Springer, 2016.
[39] D. Maccagnola, E. Messina, Q. Gao, and D. Gilbert, "A machine learning approach for generating temporal logic classifications of complex model behaviours," in *Proc. Winter Simulation Conf. (WSC)*, Dec. 2012, pp. 1–12.
[40] J. Ezpeleta, J. Fabra, and P. Álvarez, "On the use of log-based model checking, clustering and machine learning for process behavior prediction," in *Proc. 5th Int. Conf. Social Netw. Anal., Manage. Secur. (SNAMS)*, Oct. 2018, pp. 209–214. doi: 10.1109/SNAMS.2018.8554490.
[41] R. V. Borges, A. D. Garcez, and L. C. Lamb, "Integrating model verification and self-adaptation," in *Proc. IEEE/ACM Int. Conf. Automat. Softw. Eng.*, 2010, pp. 317–320.
[42] V. Tulsian, A. Kanade, R. Kumar, A. Lal, and A. V. Nori, "MUX: Algorithm selection for software model checkers," in *Proc. 11th Work. Conf. Mining Softw. Repositories*, 2014, pp. 132–141.
[43] R. Behjati, M. Sirjani, and M. N. Ahmadabadi, "Bounded rational search for on-the-fly model checking of LTL properties," in *Fundamentals Softw. Engineering—FSEN* (Lecture Notes in Computer Science), vol. 5961, F. Arbab and M. Sirjani, Eds. Berlin, Germany: Springer, 2009.
[44] E. Clarke, A. Gupta, J. Kukula, and O. Strichman, "SAT based abstraction-refinement using ILP and machine learning techniques," in *Computer Aided Verification*. Berlin, Germany: Springer, 2002.
[45] C. S. Păsăreanu and M. Bobaru, "Learning techniques for software verification and validation," in *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*. Berlin, Germany: Springer, 2012.
[46] *Machine Learning Methods for Model Checking in Continuous Time Markov Chains*. Accessed: Oct. 2014. [Online]. Available: http://www.cs.ox.ac.uk/seminars/1195.html
[47] L. Bortolussi, D. Milios, and G. Sanguinetti, "Machine learning methods in statistical model checking and system design—Tutorial," in *Runtime Verification*. Cham, Switzerland: Springer, 2015.
[48] T. Brázdil *et al.*, "Verification of Markov decision processes using learning algorithms," *Comput. Sci.*, vol. 8837, nos. 11–12, pp. 98–114, 2014.
[49] L. Bortolussi and S. Silvetti, "Bayesian statistical parameter synthesis for linear temporal properties of stochastic models," in *Proc. Int. Conf. Tools Algorithms Construct. Anal. Syst.* Cham, Switzerland: Springer, 2018, pp. 396–413.
[50] L. Bortolussi and G. Sanguinetti, "Learning and designing stochastic processes from logical constraints," in *Proc. Int. Conf. Quant. Eval. Syst.* New York, NY, USA: Springer-Verlag, 2013, pp. 89–105.
[51] A. de Matos Pedro, P. A. Crocker, and S. M. de Sousa, "Learning stochastic timed automata from sample executions," in *Proc. Int. Conf. Leveraging Appl. Formal Methods*. New York, NY, USA: Springer-Verlag, 2012.
[52] L. Belzner and T. Gabor, "Bayesian verification under model uncertainty," in *Proc. IEEE/ACM Int. Workshop Softw. Eng. Smart Cyber-Phys. Syst.*, May 2017, pp. 10–13.
[53] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini, "Approximate model checking of stochastic hybrid systems," *Eur. J. Control*, vol. 16, no. 6, pp. 624–641, 2010.
[54] A. Abate, J. P. Katoen, and M. Prandini, "A two-step scheme for approximate model checking of stochastic hybrid systems," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 4519–4524, 2011.

[55] P. Quaglia and S. Schivo, "Approximate model checking of stochastic COWS," in *Proc. Int. Symp. Trustworthy Global Comput.*, 2010, pp. 335–347.
[56] S. Basu, A. P. Ghosh, and R. He, "Approximate model checking of PCTL involving unbounded path properties," in *Proc. 11th Int. Conf. Formal Eng. Methods (ICFEM)*, Rio de Janeiro, Brazil, Dec. 2009, pp. 326–346.
[57] B. Herd, S. Miles, P. McBurney, and M. Luck, "Verification and validation of agent-based simulations using approximate model checking," in *Multi-Agent-Based Simulation XIV*. Berlin, Germany: Springer, 2013.
[58] Y. Feng and L. Zhang, "Precisely deciding CSL formulas through approximate model checking for CTMCs," *J. Comput. Syst. Sci.*, vol. 89, pp. 361–371, Nov. 2017.
[59] D. Henriques, J. G. Martins, P. Zuliani, A. Platzer, and E. M. Clarke, "Statistical model checking for Markov decision processes," in *Proc. 9th Int. Conf. Quant. Eval. Syst.*, Sep. 2012, pp. 84–93.
[60] V. Mysore and B. Mishra, *Algorithmic Algebraic Model Checking III: Approximate Methods*. Amsterdam, The Netherlands: Elsevier, 2006.
[61] W. Jamroga, "Model checking strategic ability why, what, and especially: How?" in *Proc. 25th Int. Symp. Temporal Represent. Reasoning*, 2018, pp. 3:1–3:10.
[62] W. Jamroga, M. Knapik, and D. Kurpiewski, "Fixpoint approximation of strategic abilities under imperfect information," in *Proc. Proc. 16th Conf. Auto. Agents MultiAgent Syst. (AAMAS)*, 2017, pp. 1241–1249.
[63] J. M. Davoren, T. Moor, R. P. Goré, V. Coulthard, and A. Nerode, "On two-sided approximate model-checking: Problem formulation and solution via finite topologies," in *Proc. Int. Symp. Formal Techn. Real-Time Fault-Tolerant Syst. Formal Modeling Anal. Timed Syst.* Grenoble, France: FTRTFT, Sep. 2004, pp. 52–67.
[64] K. Hamaguchi, K. Masuda, and T. Kashiwabara, "Approximate model checking using a subset of first-order logic," *IPSJ Trans. Syst. LSI Des. Methodol.*, vol. 3, pp. 268–282, Jan. 2010.

WEIJUN ZHU received the Ph.D. degree in computer science from Xidian University, China, in 2011. He is currently an Associate Professor with Zhengzhou University, China. His research interests include AI & machine learning, and formal methods & model checking.

HUANMEI WU received the Ph.D. degree in computer and information science from Northeastern University, USA, in 2005. She is currently an Associate Professor with IUPUI, USA. Her research interests include machine learning, data mining, imaging processing, and pattern recognition.

MIAOLEI DENG received the Ph.D. degree in computer science from Xidian University, China, in 2011. He is currently an Associate Professor with the Henan University of Technology, China. His research interests include AI & machine learning.