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 unacceptable 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.
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, $10^{20}$ 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.

![Diagram](image_url)
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.

![Kripke structure K=K0](image)

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

**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

### 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 systematic models K, a LTL formulas f, and their 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 m1 times, and a training set containing m1 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.

### TABLE I

<table>
<thead>
<tr>
<th>Length of formulas, i.e., L</th>
<th>Average running time (t1) of NuSMV for one pair of Kripke structure and formula</th>
<th>Average running time (t2) of NuXMV for one pair of Kripke structure and formula</th>
<th>ML algorithms</th>
<th>Average predictive time (t3) of the new method based on ML for one record (s)</th>
<th>t1/t2</th>
<th>t3/t1</th>
</tr>
</thead>
<tbody>
<tr>
<td>L=25</td>
<td>0.0150s</td>
<td>0.0150s</td>
<td>RF</td>
<td>0.000046</td>
<td>326</td>
<td>326</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BT</td>
<td>0.000037</td>
<td>405</td>
<td>405</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DT</td>
<td>0.000045</td>
<td>333</td>
<td>333</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LR</td>
<td>0.000045</td>
<td>333</td>
<td>333</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RF</td>
<td>0.000052</td>
<td>4370769</td>
<td>497500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BT</td>
<td>0.000044</td>
<td>5165455</td>
<td>5879545</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DT</td>
<td>0.000051</td>
<td>4456471</td>
<td>5072549</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LR</td>
<td>0.000051</td>
<td>4456471</td>
<td>5072549</td>
</tr>
</tbody>
</table>

| L=500                     | 227.28s                                                                         | 258.70s                                                                         | RF            | 0.000051                                                                        | 4456471 | 5072549 |

### TABLE II

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>RF</th>
<th>BT</th>
<th>DT</th>
<th>LR</th>
<th>t1 / t2</th>
<th>t3 / t1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction Accuracy</td>
<td>0.8918</td>
<td>0.9052</td>
<td>0.8560</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Running time per record (in second)</td>
<td>0.000049</td>
<td>0.000036</td>
<td>0.000033</td>
<td>0.000032</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUC</td>
<td>0.7244</td>
<td>0.9792</td>
<td>0.6138</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specificity (TNR)</td>
<td>0.996</td>
<td>0.991</td>
<td>0.994</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensitivity (TPR)</td>
<td>0.454</td>
<td>0.546</td>
<td>0.234</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.964</td>
<td>0.937</td>
<td>0.897</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prediction Accuracy</td>
<td>0.8918</td>
<td>0.9052</td>
<td>0.8560</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Running time per record (in second)</td>
<td>0.000049</td>
<td>0.000036</td>
<td>0.000033</td>
<td>0.000032</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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, $200 \times 50 = 10000$ 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 $100 \times 50 = 5000$ 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 II.

--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.

Example 1 Fig.2 illustrates an example on Kripke structure $K_0$. $K_0$ 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 $s_2$ or state $s_3$, and so on. $K_0$ can be represented with a string 00001001001110101122123403243. 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 $K_0=\\text{“00001001001110110122123403243”}$ and $f_1=\\text{“!X((!F((!p&q|r)U(p&q|r)))U(F(p&q&r)))”}$, NuXMV model checking is run and returns it model checking result, i.e., “true (yes)”, indicating $K_0$ satisfy $f_1$. Therefore, the three fields $K$, $f$, $r$ of this record are “00001001001110110122123403243”, “!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 $K_0=\\text{“00001001001110110122123403243”}$ and $f_2=\\text{“!X((!F((!p&q&r)))U((p&q&r)))”}$, its model checking result, i.e., “false (no)”, is reported by NuXMV, which indicates $K_0$ does not satisfy $f_2$. Therefore, the three fields $K$, $f$, $r$ of this record are “00001001001110110122123403243”, “!X((!F((!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 $A_1$ containing 405 records is obtained, where the length of each formula is 25, as well as a data set $A_2$ containing 405 records is obtained where the length of each formula is 500. It should be noted that $A_1$ and $A_2$ use the same Kripke structures.

Furthermore, a data set $A_3$ containing 5000 records is obtained where the length of each formula is 500, as well a data set $A_4$ containing 10000 records is obtained where the length of each formula is 500. It should be noted that $A_3$ and $A_4$ 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 $A_5$ including 10000 records. Similar with $A_3$ and $A_4$, $A_5$ has some randomly generated sub-formulas and the length of each formula is 500. Differ from $A_3$ and $A_4$, $A_5$ is not a sub-set of the data set originated from the above 10000 groups model checking.

$A_1$, $A_2$, $A_3$, $A_4$ and $A_5$ provide the raw data for our Graph lab experiments.

D. EXPERIMENTAL RESULTS

1) THE EXPERIMENTAL RESULTS ON $A_1$ AND $A_2$

First, Fig.3 illustrates some comparisons of predictive accuracy and average predictive time for one record, when $A_1$, $A_2$ 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.

### TABLE III

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

As shown in table I, 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 I. 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 II and table III.

This time, as illustrated in table II, 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 III.

### 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.
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).
FIGURE 5. (continue) (chromatic figure) Comparison of performance of ML algorithms with the same values of hyper-parameters (seed & fraction) \((L=500 \_ A3 \text{ is used})\). (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).
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).
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).
Second, as shown in table II (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.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>RF</th>
<th>BT</th>
<th>DT</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction Accuracy</td>
<td>0.950</td>
<td>0.970</td>
<td>0.930</td>
<td>1</td>
</tr>
<tr>
<td>Running time per record (in second)</td>
<td>0.000035</td>
<td>0.000005</td>
<td>0.000038</td>
<td>0.000035</td>
</tr>
<tr>
<td>AUC</td>
<td>0.9970</td>
<td>0.9983</td>
<td>0.8040</td>
<td>1</td>
</tr>
<tr>
<td>Specificity (TNR)</td>
<td>0.995</td>
<td>0.996</td>
<td>0.996</td>
<td>1</td>
</tr>
<tr>
<td>Sensitivity (TPR)</td>
<td>0.764</td>
<td>1</td>
<td>0.611</td>
<td>1</td>
</tr>
<tr>
<td>Precision</td>
<td>0.968</td>
<td>0.981</td>
<td>0.970</td>
<td>1</td>
</tr>
</tbody>
</table>

As shown in table III, 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

![Graph](https://example.com/graph.png)

It should be noted that, this max accuracy of 100% is obtained with 492 test records in our experiments, so that the actual max accuracy \( p_{\text{max}} \) is not less than \( 492/(492+1)=492/493=99.8\% \). In other words, \( p_{\text{max}} \in [0.998, 1] \) holds.
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.

### TABLE VI

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>RF</th>
<th>BT</th>
<th>DT</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction Accuracy</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

### TABLE VII

<table>
<thead>
<tr>
<th>The distinguishing feature of this type of studies</th>
<th>The distinguishing feature of the new method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Studies Type 1) deal with some SAT problems rather than MC ones</td>
<td>address the key MC problem directly</td>
</tr>
<tr>
<td>Studies Type 2) ML and MC cooperate for a third party</td>
<td>perform ML for MC</td>
</tr>
<tr>
<td>Studies Type 3) &amp; 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</td>
<td>The employed ML has changed the trunk of MC core engine, and the new method is based on data training &amp; prediction rather than state space exploration, so that both the state explosion problem and non-polynomial complexity are avoided</td>
</tr>
</tbody>
</table>

### TABLE VIII

<table>
<thead>
<tr>
<th>The existing MC algorithms</th>
<th>The new method</th>
</tr>
</thead>
<tbody>
<tr>
<td>State explosion problem</td>
<td>Can be relieved instead of avoided</td>
</tr>
<tr>
<td>Non-polynomial time complexity</td>
<td>They are suffering from it</td>
</tr>
<tr>
<td>Before real customers use it</td>
<td>A model checker is need to be developed</td>
</tr>
<tr>
<td>Applied range</td>
<td>Very wide</td>
</tr>
<tr>
<td>Can counterexamples be generated?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

![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.

Third, as shown in Table I, 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 II (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.
FIGURE 7. (continue) (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.

This is LTL model checking

(sweet) fruit

(root) trunk

high accuracy

ML results

MC results

Now, let's do something different

This is ML binary classification

(fruit)

(root) trunk

ML core

algorithm

slowly

quickly

state space exploration technique

The classical computing theory including automata theory...

ground and soil

statistics & computational learning theory

induction from data

step 1: cut

step 2: grafting

Now, I am born. My name is approximate LTL model checking.

(sweet) fruit

(root) trunk

high accuracy

ML results

MC results

Step 3: they're hooked. So...

It's a piece of cake! "I need an answer, yes or no?" MC always asks herself. Yes, it's her job. Now, ML can manage it. He always says zero or one. Of course, it's his business. Let's assume for a moment that one is yes, and zero is no. So, he's answered her question.

Wait, one question, how to connect them?

induction from data

statistics & computational learning theory

ground and soil

step 1: cut

root
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 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 II(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 criterion 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 II(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.

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 same this 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 IV and table V, 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 V.

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 VI, indicating the performance is not
acceptable at all! In fact, the combination of table IV and table VI 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 $m \times n$ 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 testing 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 IV and table VI.

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.

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][37][38]. In the strict sense, these works relate to machine learning, not model checking. In fact, SAT denotes the 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], [47], [48], [49], [50], [51] and [52] for more details. And no more information will be detailed here, due to the limitation of space.

Table VII 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 VIII. 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 VII 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. CONCLUSIONS

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


This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2942762, IEEE Access


W. Zhu, et al.: LTL model checking based on Binary Classification of Machine Learning

Weijun Zhu received his Ph.D. degree in computer science from Xidian University, China, in 2011. Currently, he is an associate professor at Zhengzhou University, China. His research interests include AI & machine learning, and formal methods & model checking.

Huanmei Wu received her Ph.D. degree in Computer and Information Science from Northeastern University, USA, in 2005. Currently, she is an associate professor at IUPUI, USA. Her research interests include machine learning, data mining, imaging processing and pattern recognition.

Miaolei Deng received his Ph.D. degree in computer science from Xidian University, China, in 2011. Currently, he is an associate professor at Henan University of technology, China. His research interests include AI & machine learning.