Int J Software Informatics, Volume 10, Issue 4 (2016), pp. 000–000
International Journal of Software and Informatics, ISSN 1673-7288
©2016 by ISCAS. All rights reserved.

E-mail: ijsi@iscas.ac.cn
http://www.ijsi.org
Tel: +86-10-62661048

DOI: 10.21655/ijsi.1673-7288.00235

AL-SMC: Optimizing Statistical Model Checking by Automatic Abstraction and Learning

Kaiqiang Jiang, Ping Huang, Hui Zan and Dehui Du

(Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China)

Abstract Statistical Model Checking (SMC), as a technique to mitigate the issue of state space explosion in numerical probabilistic model checking, can efficiently obtain an approximate result with an error bound by statistically analysing the simulation traces. SMC however may become very time consuming due to the generation of an extremely large number of traces in some cases. Improving the performance of SMC effectively is still a challenge. To solve the problem, we propose an optimized SMC approach called AL-SMC which effectively reduces the required sample traces, thus to improve the performance of SMC by automatic abstraction and learning. First, we present property-based trace abstraction for simplifying the cumbersome traces drawn from the original model. Second, we learn the analysis model called Prefix Frequency Tree (PFT) from the abstracted traces, and optimize the PFT using the two-phase reduction algorithm. By means of the optimized PFT, the original probability space is partitioned into several sub-spaces on which we evaluate the probabilities parallelly in the final phase. Besides, we analyse the core algorithms in terms of time and space complexity, and implement AL-SMC in our Modana Platform to support the automatic process. Finally we discuss the experiment results for the case study :energy-aware building which shows that the number of sample traces is effectively reduced (by nearly 20% to 50%) while ensuring the accuracy of the result with an acceptable error.

Key words: statistical model checking; statistical abstraction; learning; principal components analysis; prefix frequency tree

Jiang KQ, Huang P, Zan H, Du DH. AL-SMC: Optimizing statistical model checking by automatic abstraction and learning. ***Int J Software Informatics*****, Vol.10, No.4 (2016): 000–000. http://www.ijsi.org/1673-7288/10/235.htm**

1 Introduction

Statistical Model Checking (SMC)[23] is a simulation based efficient technique to evaluate the probability that a system model satisfies a given property, or to verify whether a system model satisfies a given property with a specified probability threshold. The former is known as **Quantitative analysis** including Simple Sampling Plan (SSP)[23], Sequence Probability Ratio Test (SPRT)[29,30] and

This work was supported by NSFC (Grant No.61472140, 61202104) and NSF of Shanghai (Grant No. 14ZR1412500).
Corresponding author: Dehui Du, Email: dhdu@sei.ecnu.edu.cn
Received 2016-07-30; Revised 2016-12-12; Accepted 2017-01-07.

Bayesian Hypothesis Testing (BHT)[19], which will give a real-value answer (between 0 and 1), i.e. $M \models P_{=?}(\phi')$. The latter is **Qualitative analysis** including Approximate Probabilistic Model Checking (APMC)[15] and Bayesian Interval Estimation (BIE)[31], which will give a boolean answer, i.e. $M \models P_{\geqslant\theta}(\phi')$. Compared to numerical (probabilistic) model checking[1] that obtains an accurate result by means of exhaustive exploration of the state space, SMC computes an approximate result as well as an error bound by combining Monte-Carlo simulation and statistical techniques. The samples used in SMC is called **Traces** which are drawn from the system model by simulation tools. It means we need not know more details of the system than its simulation traces, and some difficulties (e.g. state space explosion) in verifying complex systems are effectively mitigated. Therefore, SMC is practically useful in analysing stochastic, hybrid, heterogeneous, even black-box systems (typically like Cyber-physical Systems (CPS)[22]), and it is also the only option in most cases.

However, SMC may encounter the performance bottleneck in some cases, for example, using BIE to check the property of which the satisfaction probability is close to 0.5, using SPRT/BHT to check the property with a threshold θ close to its real probability, checking the property of rare event with high precision, etc. The performance of SMC is mainly affected by two factors: (i) the simulation time of executing a single trace, which is a quite time-consuming process for most cases; (ii) the number of traces required for termination of SMC algorithm. The time spent in statistical process turns out to be a small part of the total time in SMC (i.e. 10% of the total time[31]). As usual, the simulation time depends upon the efficiency of the simulation tool and the complexity of the system model. As a result, the key to improve SMC performance is how to reduce the number of sample traces, which is still a challenge problem.

To address the challenge, we propose an optimized SMC approach based on automatic **Abstraction** and **Learning** (**AL-SMC**, for short). The abstraction technique is used to derive the concise, abstract traces from the cumbersome, original ones; and the learning technique is used to build a more appropriate analysis model, namely, Prefix Frequency Tree (PFT), for evaluating the probability in a more efficient way. Our approach is partially inspired by Ref. [25] but quite different in nature. The major differences lie in (i) our approach focuses on abstracting CPS traces that contains both continuous and discrete variables; (ii) our approach uses PFT as the final analysis model for partitioning the original probability space into several sub-spaces (instead, PFT is used as an intermediate model in Ref. [25]); (iii) our approach improves the SMC performance by efficiently evaluating the probability on each sub-space in parallel (instead, Ref. [25] directly analyse the abstracted probabilistic finite automata constructed based on PFT).

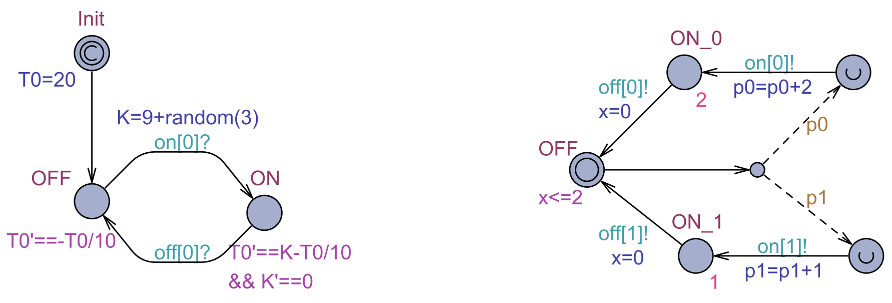
The main contributions of our work are as follows: (i) we present property-based trace abstraction to obtain concise traces; (ii) we partition the probability space into several balanced sub-spaces by learning PFT ; (iii) we present the core algorithms for abstraction and learning, and also implement the approach in our Modana platform to facilitate the automatic process.

The rest of this paper is organized as follows. Section 2 presents the preliminaries and overview including a motivating example, basic idea and the framework of AL-

SMC. Section 3 discusses the core algorithms of AL-SMC in details. Section 4 presents our implementation and the experimental results on a case study. Section 5 discusses related work and in section 6, we conclude the work and discuss the future work.

2 Preliminaries and Overview

In this section, we briefly introduce Stochastic Hybrid Automata (SHA)[8] and Probabilistic Bounded Linear Temporal Logic (PBLTL)[7] used in this paper by a motivating example. And then, we verify the motivating example with SMC to introduce the problem. Then we present the basic idea of our optimized approach as well as the framework of AL-SMC to solve the problem.



(a) SHA template for room (b) SHA template for heater

Figure 1. Modeling of motivating example.

Motivating example. A simple 2-room example will be introduced for illustration of SHA as well as AL-SMC later. Given that there are two rooms sharing one heater. Without considering other factors (e.g. environment temperature), the room temperature changes with the equation $T' = K - \frac{T}{10}$. T' is the change rate of room temperature; $K = 9 + random(3)$ represents a random heating capacity, so K equals a real value between 9 and 12 when the room gains the heater and otherwise $K = 0$. The time delays at the locations ON_0, ON_1 follow exponential distribution $exp(\lambda)$ ($\lambda = 2$ for ON_0, $\lambda = 1$ for ON_1). And the strategy for deciding which room to be heated is described by a discrete probabilistic branch ($p0, p1$ as the weights of $room0, room1$). The probability that the heater chooses $ith\ room$ is $p_i / \sum p_i$. The SHA models of the 2-room example are shown in Fig. 1.

Definition 1. *Probabilistic Bounded LTL (PBLTL)* The requirement constraints of the system are specified with the property PBLTL $P_{=?}(\phi')$ where ϕ' is a BLTL property. The syntax of BLTL is given by the following grammar:

$$\varphi ::= y \sim v \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi_1 \mid \phi_1 U^{\leqslant t} \phi_2$$

where $\sim \in \{\geqslant, \leqslant, =\}$, $y \in SV$ (SV is set of all variables), $v \in \mathbb{Q}$, $t \in \mathbb{Q}_{\geqslant 0}$. As usual, we define additional temporal operators such as the operator "eventually within time t" which is defined as $F^{\leqslant t}\phi = True\ U^{\leqslant t}\phi$, or the operator "always up to time t" which is defined as $G^{\leqslant t}\phi = \neg F^{\leqslant t}\neg\phi$. More details about PBLTL can be found in Ref. [31].

For example, we can evaluate the probability that the temperature difference between two rooms is too large, using the following PBLTL property $\phi = P_{=?}(\phi')$

$$P_{=?}(F^{\leqslant 48}\ T0 > 31\ \wedge\ T1 < 4)$$

ϕ' indicates that the temperature of $room0$ exceeds 31 meanwhile the temperature of $room1$ declines below 4. Next, we will present the basic idea of our AL-SMC optimization with the motivating example.

2.1 Basic idea

We apply the state-of-the-art quantitative SMC algorithm - BIE[31,20] to evaluate the probability for a quantitative property ϕ. Figure 2 shows an overall estimation on the number of sample traces generated by the BIE algorithm.

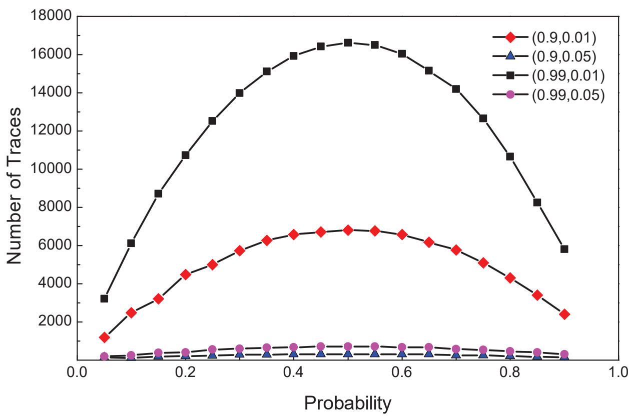


Figure 2. Simulation traces for different probabilities using BIE.

We found that it needs the most traces when the probability is close to 0.5, while, the traces are drastically reduced when the probability approaches to 0 or 1. So, we can evaluate the probabilities on several sub-spaces instead of the original space. Given that the original probability space Ω is partitioned into m sub-spaces $\Omega_1, \cdots, \Omega_m$ with the probabilities: $p_1, \cdots, p_m$; $Trs(p_i)$ represents the number of sample traces required for evaluating p_i. From Fig. 2, it is derived that $Trs(P) < Trs(P')$ if $P < P' < 0.5$. Consequently for m sub-spaces, $Trs(p_i) < Trs(p)$ such that $p = \sum_{i=1}^{m} p_i$ (if the original probability $p \leqslant 0.5$). As the evaluation on m sub-spaces are performed in parallel, the actual number of traces is $Trs(p') = \max(Trs(p_1), \cdots, Trs(p_m)) < Trs(p)$. That is, the number of traces for evaluating p will be decreased and depends on the maximum of trace number for p_i of each sub-space theoretically.

We check the illustrative property ϕ using BIE ($\delta = 0.02$, $c = 0.95$) algorithm. And the experimental results show that its satisfaction probability is 0.465 and the number of sample traces is 2885. The probability is close to 0.5, which means the number of sample traces can be theoretically reduced according to the above idea.

However, the main difficulty is how to find a set of appropriate probability sub-spaces. As shown in Fig. 3, the probability distribution is not balanced in most cases. To solve the problem, we have to rebuild an equivalent model in another dimensional space instead of directly partitioning the space in temporal dimension. According to the basic idea of our approach, we present the framework **AL-SMC** which mainly focuses on how to rebuild an abstract analysis model (i.e. PFT) probabilistically equivalent to the original SHA models by means of trace abstraction.

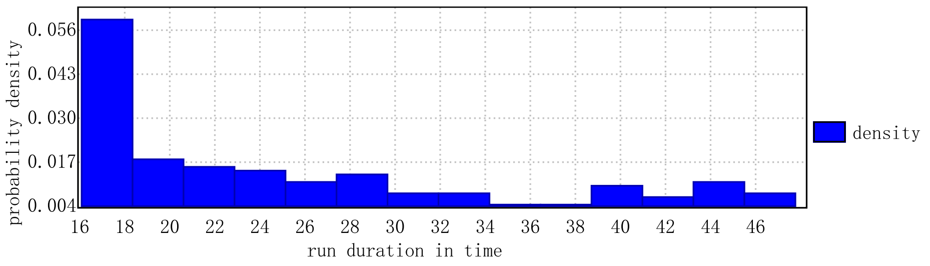
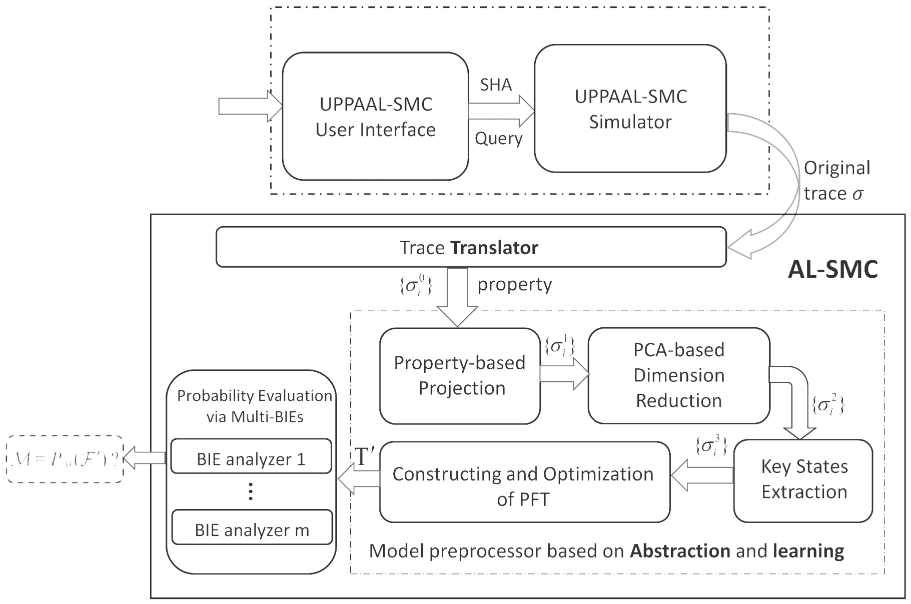


Figure 3. The probability density distribution for motivating example satisfies ϕ'.



UPPAAL-SMC User Interface
SHA
Query
UPPAAL-SMC Simulator
Original trace σ
Trace Translator
AL-SMC
{σ_i^0}
property
Property-based Projection
{σ_i^1}
PCA-based Dimension Reduction
{σ_i^2}
Key States Extraction
{σ_i^3}
Constructing and Optimization of PFT
T'
Probability Evaluation via Multi-BIEs
BIE analyzer 1
BIE analyzer m
M ⊨ P_θ(F')?
Model preprocessor based on Abstraction and learning

Figure 4. Framework of AL-SMC.

2.2 Framework of AL-SMC

The **AL-SMC** framework is depicted in Fig. 4. Initially, the original traces are obtained by the simulator of UPPAAL-SMC which supports to model the system based on SHA. Next, **Trace Translator** is employed to translate the original traces to normal traces which is the input of AL-SMC. The abstract process is performed by three components: **Property-based Projection**, **PCA-based Dimension Reduction** and **Key States Extraction** which are employed to obtain more abstract traces (from σ^0 to σ^3). By this way, the sample traces are reduced to some extent. Next, **PCA-based Dimension Reduction** is employed to partition the probabilistically equivalent model based on the abstract traces, which adopts a classic machine learning method called Principal Component Analysis (PCA)[10]. With the help of PCA-based dimension reduction, the more abstract traces (σ^3) are generated. Subsequently, the key part of AL-SMC **Building and Optimization of PFT** adopts the learning technique to build and optimize PFT models. Once the PFT models are obtained, the **Probability Evaluation via Multi-BIEs** invokes several BIE analysers to evaluate the probability in parallel.

3 Abstraction and Learning-based SMC (AL-SMC)

In this section, we first discuss core algorithms for **property-based trace abstraction**, **construction and optimization of PFT** and **probability evaluation via multi-BIEs**. Further, we briefly discuss the time and space complexity of each algorithm.

3.1 Property-based trace abstraction

Property-based trace abstraction is composed of three phases: (i) dispersing continuous variables with property-based projection; (ii) dimension reduction for states with the PCA-based dimension reduction technique; (iii) extracting key states with two parameters. The aim of property-based trace abstraction is to simplify the traces by projection and abstraction techniques.

3.1.1 Property-based projection

As we known, each trace generated by UPPAAL-SMC simulator contains many states which is composed of discrete and continuous variables. The discrete variables denote locations and the continuous variables denote continuous behavior of the system, such as energy consumption, temperature change, etc. However, only the states with discrete variables changing have significantly influences on continuous behavior of the system. For this kind of states, we call it discrete states. Besides, we find the number of simulation traces will reduce when the probability of the original model is less than 0.5. But if the probability of original model is bigger than 0.5, the probability of sub-model may approximate to 0.5. Therefore, a small number of traces is needed to estimate the probability of original spaces.

Before using Algorithm 1, we should determine the BLTL ϕ'. Algorithm 1 performs projection on the original traces σ^0 according the property BLTL ϕ'. The preprocess of the algorithm is to determine the form of ϕ' with the following equation:

$$\phi'_{smc} = \begin{cases} \phi', & p_{test} \leqslant 0.5 \\ \neg(\phi'), & p_{test} > 0.5 \end{cases} \qquad p_{test} = \Big(\sum_{i=1}^{100} b_i\Big)/100 \tag{1}$$

where b_i denotes whether the ith trace satisfies property ϕ', p_{test} is the probability of original space evaluated with AL-SMC. We can ensure p_{test} less than 0.5 using this estimation.

In Algorithm 1, $SV = L \cup X$ denotes the set of discrete variables of states, and σ^0 denotes an original trace which consists of k dimensional states ($k = \#(SV)$). First, discrete states are only kept in each trace to simplify the cumbersome original traces. Then, we add two dimensions $sChk, tChk$ to each state of the trace. $sChk$

Algorithm 1 Property-based projection

```
Require: BLTL property $\phi'$, Original trace $\sigma^0$
Ensure: New trace $\sigma^1$ after projection
1: $T := \{0\}$   //the set of discrete states
2: $\sigma^1 := \emptyset$
3: for all $\nu_i \in \sigma^0$ do
4:     if $\pi_L(\nu_i) \neq \pi_L(\nu_{i-1})$ then
5:         $T := T \cup \{t(\nu_i)\}$   //$\pi_L(\nu_i)$ is the set of discrete variables of state
6: for all $t_j \in T$ do
7:     $(sChk_j, tChk_j) := (0,0)$
8:     if $\sigma^0(t_{i-1}, t_i) \models \phi'$ then
9:         $sChk_j := 1$   //whether $state_i$ satisfies $\phi'$
10:    if $tChk_{j-1} = 1$ or $sChk_j = 1$ then
11:        $tChk_j := 1$
12:    $\nu'_j := \pi_L(\nu_j) \bowtie (sChk_j, tChk_j)$   //adding two dimensions to compose new
       state
13:    $\sigma^1 := \sigma^1 \cup \{\nu'_j\}$   //adding new state to new trace
14: return $\sigma^1$
```

denotes whether the state satisfies ϕ' and $tChk$ denotes whether the trace satisfies ϕ'. Finally, all continuous variables are projected to $sChk$ and $tChk$, and then new states with $k+2$ dimensions are obtained to compose the new trace σ^1. By this way, the property-based projection technique is employed to abstract the traces.

Figure 5 shows the trace σ^1 generated with property-based projection, which is more concise compared to the original one. Whereas the number of states is still large, so it is difficult to analyse the essential behaviour of the system. To reduce the number of states, we will further abstract σ^1 to get the trace with only discrete states.

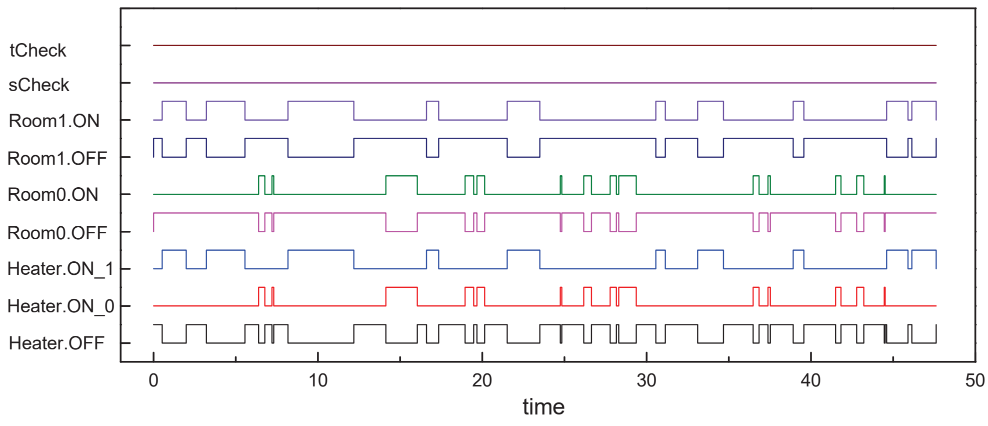


Figure 5. Trace σ^1 generated by property-based projection.

3.1.2 PCA-based dimension reduction

PCA[10] is proposed by Pearson[27] to analyse and simplify the data set for multivariate statistical analysis. The main idea of PCA is projecting the feature vectors from k_0-dimensions to k_1-dimensions ($k_0 \gg k_1$). By this way, PCA keeps the significant features of original set which make great contribution to the feature analysis. To further abstract the trace σ^1, we analyse the discrete variables of each state, and find a lot of variables are positively correlated, i.e. the combination of variables can be used to represent the key behavior of the system. Inspired by the PCA technique, we propose the PCA-based dimension reduction to observe the main behaviour of the system.

In this phase, each state can be treated as a sample of PCA, and the variables of each state can be treated as the feature of a sample. After the PCA-based dimension reduction, the dimension of feature is greatly reduced as shown in Fig. 6. We can find that each state only contains two features, which are the combination of discrete variables. So, it facilitates to extract the key states in next stage.

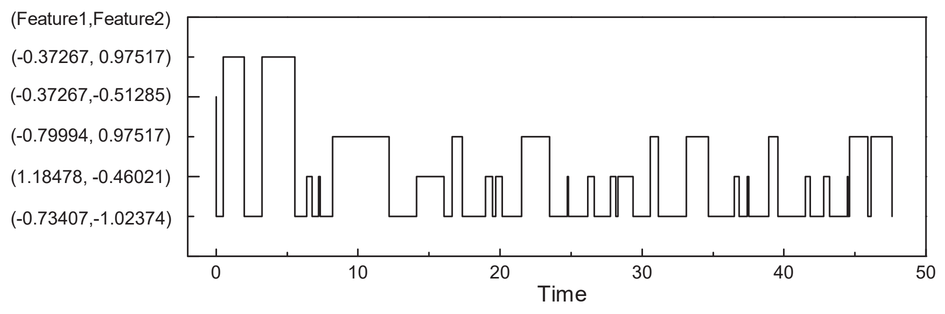


Figure 6. Trace σ^2 generated by PCA-based dimension reduction.

3.1.3 Key states extraction

Through the analysis of trace σ^2, we find that few states are useful to evaluate the probability of property ϕ. Key states should be extracted from trace σ^2 to compose a concise trace. The main purpose of key states extraction is to find key states which have great influence for evaluating the probability. The following are the main steps of key states extraction:

1. Identifying the key states where $tChk$ equals to 1. These key states directly influence the evaluation of probability.

2. Extracting the states whose features occur frequently. We define two thresholds to denote the number of extracting states: 1) the maximal number of extracting states 2) the maximal percent of extracting states in total states.

After key states extraction, we obtain the trace σ^3 as shown in Fig. 7. The behaviour of the system can be described with two key states which are two compositions of discrete variables (1001010 and 0100110) in the original traces. The first composition denotes one heater is OFF, the second one denotes the $heater$ is ON and $room0$ is heated. The analysis results show that our evaluation is correct: the heating time of $room0$ is longer and dominant during the heating process because the weight of $room0$ is higher (shown in Fig. 1).

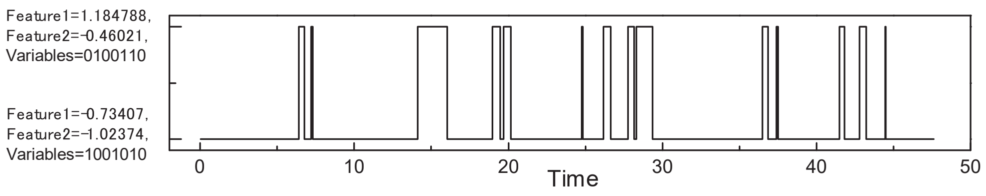


Figure 7. Trace σ^3 generated by key states extraction.

3.2 Construction and optimization of PFT

The aim of construction and optimization of PFT is to partition the original probability space into several relatively balanced sub-spaces by learning a PFT. First, a PFT is constructed based on the state set of σ^3, and then the PFT is optimized by two-phase reduction.

Definition 2. *Prefix Frequency Tree (PFT)* A PFT (multi-branch tree) $\mathbb{T}$ is a tuple (D, R, d_0) where

- D denotes the set of tree nodes. Each tree node contains three key variables (id, f, n) : id denotes a unique state, f denotes the number of traces which terminate at this node, n denotes the number of traces passing the node, and each node satisfies $f \leqslant n$.
- R denotes the relation between nodes, and each node has one parent node and many children nodes except the root node.
- $d_0 \in D$ is the root of the tree which has no parent.

PFT is inspired by the Prefix Tree Acceptor in Ref. [6] which presented the methods of stochastic grammar inference to derive grammar automata from a set of sentences. Actually, learning the probabilistic automata is equivalent to infer stochastic grammar if we deal with each concise trace as a sentence. We take the negative trace as a null sentence, i.e. the negative trace terminate at the root of PFT, and the positive trace terminates at a non-root node. By this way, we construct an abstract model PFT which is probabilistic equivalent to the original SHA model. For PFT, we suppose that:

1. $n(d_{leaf}) = f(d_{leaf})$, i.e. f and n of all the leaf nodes are equal.
2. $n(d_i) - f(d_i) = \sum_{d_t \in child(d_i)} n(d_t)$, i.e. the difference between n and f of any non-leaf nodes equals to the sum of n of all the child nodes.
3. $n(d_0) - f(d_0) = \sum_{i>0} f(d_i)$, i.e. the difference between n and f of root node equals to the sum of f of all the non-root nodes.

For the motivating example, we found that the PFT is built with 250 traces. The total number of nodes is 1025 and the number of end nodes is 46, but the number of positive traces in σ^3 is only 121. There must be some end nodes which accepts seldom traces when the positive traces are dispersed in 46 nodes (f is too small). It means the probability of the branches satisfying the property is also small.

Algorithm 2 Constructing prefix frequency tree

```
Require: The set of $\sigma^3$
Ensure: Prefix frequency tree $\mathbb{T}$
 1: $d_{temp} := d_0$   //$d_{temp}$ is the current node
 2: for all $\sigma_i^3 \in \Sigma^3$ do
 3:     $n(d_0) := n(d_0) + 1$
 4:     if $tChk_{last} = 0$ of $\sigma_i^3$ then
 5:         $f(d_0) := f(d_0) + 1$   //terminating at $d_0$ if the trace is not satisfied
 6:     else
 7:         for all $\nu_j \in \sigma_i^3$ do
 8:             if $id(d_{temp}) = id(\nu_j)$ then
 9:                 $n(d_{temp}) := n(d_{temp}) + 1$   //merging the duplicate states
10:             else if $child(d_{temp})$ contains $id(\nu_j)$ then
11:                 $n(d_{id}) := n(d_{id}) + 1$   //$d_{id}$ represents the child node of $d_{temp}$ whose id
                    equals to $id(\nu_j)$
12:                 $d_{temp} := d_{id}$   //updating the current node
13:             else
14:                 add $\nu_j$ to $child(d_{temp})$
15:                 $d_{temp} := d_{id}$
16:             if $tChk_j = 1$ then
17:                 $f(d_{temp}) := f(d_{temp}) + 1$   //the value of $f$ of the current node plus 1 if
                    reaching the last state of the trace
18: return $\mathbb{T}$ with $d_0$
```

To avoid the small probability and reduce the error of the algorithm, the PFT is reduced with two phases reduction. First, we **horizontally merge (reduction phase I)** the branches whose value of n is low, then we **vertically merge (reduction phase II)** the paths whose value of f is low.

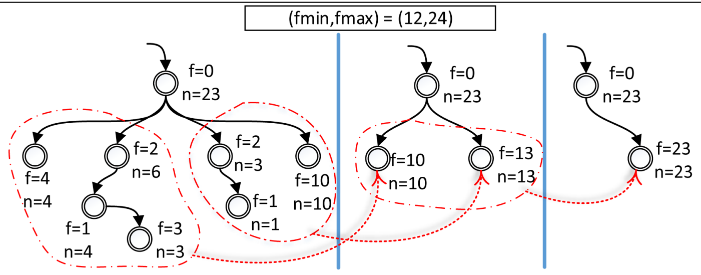
By this way, the number of positive samples in each end node is within a suitable range $(f_{\min}, f_{\max})$ which is computed with equation (2). The input parameter r denotes the reduction degree of PFT. Algorithm 2, 3 and 4 are the pseudo-codes for constructing and optimization of PFT, respectively.

$$\begin{aligned} f_{\min} &= \lfloor (r - 0.05) \cdot (n(d_0) - f(d_0)) \rfloor \\ f_{\max} &= \lfloor (r + 0.05) \cdot (n(d_0) - f(d_0)) \rfloor \end{aligned} \tag{2}$$

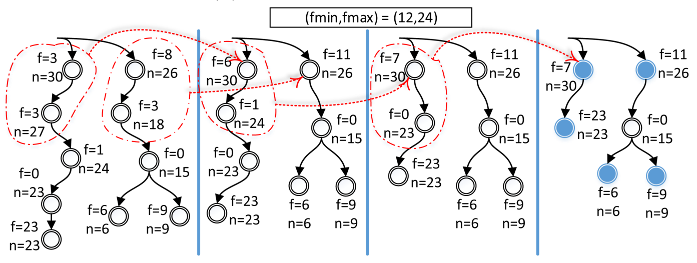
Figure 8(a) illustrates the reduction phase I. As we can see, on the left is the original tree, and the nodes marked with $n = 4, 6, 3, 10$ (less than $f_{\min}$) can be merged. First, we merge the nodes marked with $n = 3$ and $n = 10$ (the maximum and minimum value, respectively) to compose a new node marked with $n = 13$. Next, the nodes marked with $n = 4$ and $n = 6$ are merged to compose a new node marked with n=10, and this node still need to be merged because the value of n is less than $f_{\min}$. Next, the node marked with $n = 10$ and $n = 13$ are merged, due to n:$10 + 13 < f_{\max}$. Finally, the reduction tree is generated with nodes merging, as shown in the right figure.

Algorithm 3 The recursive function of PFT reduction phase I *reduce_recur*()

```
Require: The current node $d$
Ensure: Whether $n(d)$ is smaller than $f_{\min}$
 1: if $n(d) < f_{\min}$ then
 2:   return $true$
 3: $D_{need} := \emptyset$   //the set of nodes needed mergence
 4: for all $d_i \in child(d)$ do
 5:   if $reduce\_recur(d_i)$ then
 6:     add $d_i$ to $D_{need}$
 7: loop
 8:   if $size(D_{need}) = 0$ then
 9:     break      //terminating the mergence
10:   if $size(D_{need}) = 1$ then
11:     if $d \neq d_0$ and $size(child(d)) = 1$ and $f(d) + f(d_i) < f_{\max}$ then
12:       merge $d_i \in D_{need}$ to $d$   //$d_i$ is the child node of $d$
13:     else if $size(child(d)) > 1$
          and $f(min\_child(d)) + f(d_i) < f_{\max}$ then
14:       merge $d_i \in D_{need}$ to $min\_child(d)$   //$min\_child(d)$ is the node with
          minimum f value of the child nodes of $d$
15:     break
16:   merge $d_{nMax}, d_{nMin}$ in $D_{need}$
17:   delete $d_{nMax}, d_{nMin}$ from $D_{need}$   //$d_{nMax}$ is the node with maximum n
      value in $D_{need}$
18:   if $n(d_{merged}) < f_{\min}$ then
19:     add $d_{merged}$ to $D_{need}$   //$d_{merged}$ is the node through the mergence of
        $d_{nMax}$ and $d_{nMin}$
20: return $false$
```



(a) PFT reduction phase I



(b) PFT reduction phase II

Figure 8. Illustration of two-phase PFT reduction.

Figure 8(b) illustrates the reduction phase II. As we can see, the PFT reduced by reduction phase I is shown in the left. There are some end nodes with $f \leqslant f_{\min}$. First, the starting nodes in left and right branches with $f = 3$ and $f = 8$ (less than $f_{\min}$) are merged. At this time, we find that the left branch is still need to be merged (where f: $6+1 < f_{\min}$), so nodes in the left branch are merged until f:$7+23 > f_{\max}$ is satisfied. After the reduction, the number of end nodes is smaller and the values of f for each nodes are much more balanced, and it is suitable for Multi-BIEs statistical analysis in next stage.

Algorithm 4 The recursive function of PFT reduction phase II $reduce2\_recur()$

```
Require: The current node $d$
1:  $needRecur := false$   //whether recursive
2:  $d_{temp} := d$
3:  if $size(child(d)) > 1$ then
4:    $needRecur := true$
5:  else if $size(child(d)) = 1$ then
6:    repeat
7:      $d_{temp} :=$ the only child of $d_{temp}$
8:      if $size(child(d_{temp})) > 1$ then
9:        $needRecur := true$
10:       break
11:     else
12:       if $f(d) < f_{\min}$ and $f(d) + f(d_{temp}) < f_{\max}$ then
13:         $f(d) := f(d) + f(d_{temp})$   //merge $d_{temp}$ and $d$
14:         merge $d_{temp}$ to $d$
15:       else
16:         if $size(child(d_{temp})) = 1$ then
17:           $d := d_{temp}$   //update initial node $d$
18:         else
19:           break   //terminating mergence if reach the leaf node
20:   until $size(child(d_{temp})) = 0$
21: if $needRecur$ then
22:   for all $d_i \in child(d_{temp})$ do
23:     $reducetree2\_recur(d_i)$   //recursive child nodes of $d_{temp}$
```

Table 1 The comparison of PFT sizes in different phases.

PFT phase	Total nodes	End nodes	Reduction rate of nodes
PFT Construction	1025	46	4.5%
PFT Reduction I	48	24	50.0%
PFT Reduction II	12	10	83.3%

Table 1 shows that the number of nodes in the original PFT is large, but after the reduction phase I, the number of total nodes is reduced a lot and the number of end nodes is reduced a half; further, after the reduction phase II, the number of total nodes and end nodes is small enough(12 and 10, respectively). Suppose the number of end nodes divide the number of total nodes denotes **reduction rate of**

nodes. With the help of two-phases reduction, the reduction rate of nodes in PFT is improved from 4.5% to 83.3%.

3.3 Probability evaluation via multi-BIEs

$\mathbb{T}'$ denotes the PFT after reduction, which is the abstract model obtained by the probability space partition of the original model. The node with f $\geqslant 0$ is called end node. And suppose that some positive traces may not be accepted by $\mathbb{T}'$, there should be $m+1$ BIE analysers. m denotes the number of end nodes of $\mathbb{T}'$. For each trace, the corresponding BIE analyser is executed. And the SMC algorithm will terminate until all BIE analysers terminate. The process is implemented with Algorithm 5.

With AL-SMC analyzing the motivating example, the evaluation results show that the probability is 0.478 and the number of simulated traces is 819. Compared with the UPPAAL-SMC verifier, the number of simulated traces reduces obviously and the probability error is higher. But, it is still less than the half-interval coefficient $\delta = 0.02$. When the interval coverage coefficient c is increased to 0.99, the probability is 0.461 and the probability error is less than that of UPPAAL-SMC, but the number of simulated traces merely increases to 1589.

Algorithm 5 Probability evaluation via Multi-BIEs

```
Require: The prefix frequency tree after optimization $\mathbb{T}'$, BLTL property $\phi'$, half-
    interval $\delta$, the number of end nodes $m$
Ensure: Probability $p$
 1: $I := \{(x_1, \gamma_1, end_1), \dots, (x_{m+1}, \gamma_{m+1}, end_{m+1})\}$   //the set of BIE analysers
 2: $N := 0$   //number of traces
 3: while $\exists\ i \in I$ that $end_i = false$ do
 4:     $\sigma := generateSampleTrace()$   //generate one trace
 5:     $\sigma' := preprocess(\sigma)$   //obtain abstraction trace $\sigma^3$
 6:     if $correct\sigma' \models \phi'$ then
 7:         $i := findEndNode(\mathbb{T}', \sigma')$   //find the end node
 8:         if $i \leqslant 0$ then
 9:             $i := m + 1$   //add trace to $(m+1)th$ analyser
10:         $x_i := x_i + 1$
11:     $N := N + 1$
12:     if $end_i$ then
13:         $i := min(1, \dots, m+1)$ that $end_i = false$   //find an non-terminated
            analyser
14:     $p_i, \gamma_i := computeStatisticalParameter_i(x_i, N)$   //compute the probability
        and ratio for $ith$ analyser
15:     if $checkEndCondition_i(\gamma_i)$ then
16:         $end_i := true$   //terminate the $ith$ analyser
17: for all $i \in I$ do
18:     $p_i, \gamma_i := computeStatisticalParameter_i(x_i, N)$   {execute the last calculation}
19:     if $p_i \leqslant \delta$ then
20:         $p_i := x_i/N$   //$p_i$ is too small and modify it
21: return $p := \sum_{i=1}^{m+1} p_i$   //compute the final probability
```

3.4 Algorithm discussion

The models or traces in each phase (σ^0 to σ^3, $\mathbb{T}$, $\mathbb{T}'$) is probabilistically equivalent in terms of a certain property. i.e.

$$P_M(\sigma^0) = P_M(\sigma^1) = P_M(\sigma^2) = P_M(\sigma^3) = P_M(\mathbb{T}) = P_M(\mathbb{T}')$$

So the correctness of probability evaluation on the final abstract model $\mathbb{T}'$ is guaranteed. The evaluation error is affected by the multi-BIE statistical analysis.

The time and space complexity of core algorithms in AL-SMC are shown in Table 2. 1) The main function of property-based projection is to abstract traces preliminarily and its time complexity is $O(mn)$, where m denotes the length of the trace and n denotes the number of the training set. 2) The time complexity of PCA-based dimension reduction depends on the PCA algorithm. k denotes the dimensions of feature, the time complexity approximates to a constant. 3) The key states extraction only compares the frequency of features, so the time complexity is also a constant. 4) The algorithm of building PFT traverses every state of the trace, thus the time complexity is also $O(mn)$. 5) The PFT reduction algorithm I and the PFT reduction algorithm II are both recursive procedures based on multi-branch tree, so the time complexity is less than $O(d \log d)$ and the space complexity is $O(\log d)$, where d denotes the number of nodes in PFT. 6) Since the number of iterations of the probability evaluation via Multi-BIEs algorithm is unknown, we only measure the time complexity with a single iteration which mainly contains two parts: (i) the time of searching terminal node is $\log d$; (ii) the time of statistical analysis with BIE is $O(i)$, so the total time of this algorithm (one iteration) is $O(\log d + i)$ and i depends on the integration steps in BIE algorithm.

Analysis results show that the procedure of trace generating consumes more time than BIE statistical analysis, while the property-based projection and the establishment, optimization of PFT only consume small number of traces. So, the procedure of abstraction and learning have few effects on the efficiency of AL-SMC. In short, our AL-SMC is more efficient than classic BIE algorithm.

Table 2 The time and space complexity of the algorithms in AL-SMC.

Algorithm phase	Time	Space
Property-based projection	$O(mn)$	$O(1)$
PCA-based dimension reduction	$O(min(k^3, n^3))$	$O(k^2)$
Key states extraction	$O(1)$	$O(1)$
PFT constructing	$O(mn)$	$O(1)$
PFT Reduction I	$O(d \log d))$	$O(\log d)$
PFT Reduction II	$O(d \log d)$	$O(\log d)$
Probability evaluation (Single iteration process)	$O(\log d + i)$	$O(1)$

4 Implementation and Case Study

The AL-SMC framework has been implemented in our Modana platform[3] (https://github.com/ECNU-MODANA/AL-Modana.git) which is a modeling and analysis platform for CPS. Further, the core algorithms of AL-SMC are employed to analyse several experiments. In this section, we compare the efficiency and accuracy between **AL-SMC verifier** and **UPPAAL-SMC verifier** with a CPS benchmark: **energy-aware building**[7]. Besides, the probability error of AL-SMC is also discussed and a feasible solution to reduce probability error is proposed.

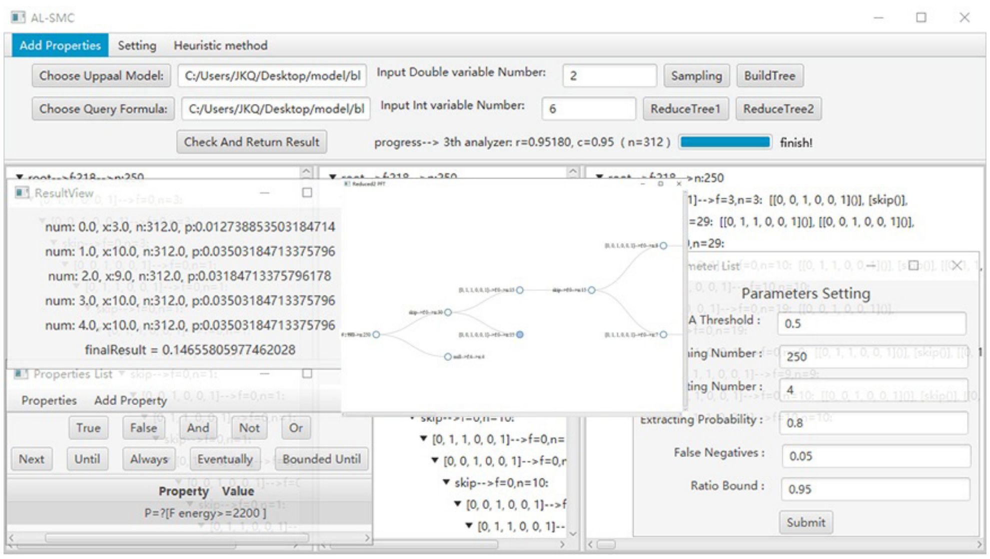
4.1 AL-SMC implementation

In our previous work, we have implemented Modana platform which supports an integrated modeling and verifying environment for CPS. Figure 9 shows the user interface of AL-SMC verifier which facilicates an automatic process of abstraction and learning.

User should customize three input parameters (ν, ω, r), where

1. ν denotes the size of the training set.
2. ω determines the degree of trace abstraction which contains three thresholds: (i) the threshold in PCA-based dimension reduction; (ii) the number of extracted states during key states extraction process; (iii) the sum of probabilities of the extracted states.
3. r denotes the reduction degree of PFT and determines $f_{\min}$ and $f_{\max}$ used in Algorithm 3 and 4.

With different parameter settings, the different PFT are builded. For example, if it needs more traces for learning and abstraction, the value of ν should be increased, or the thresholds of ω is increased to remain more states in each trace. By these ways, a more complex PFT can be obtained. In addition, increasing the value of r leads to decrease the number of end nodes of PFT, which is crucial for the probability evaluation phase.



C:/Users/JKQ/Desktop/model/bl

C:/Users/JKQ/Desktop/model/bl

Input Double variable Number: 2

Input Int variable Number: 6

progress--> 3th analyzer: r=0.95180, c=0.95 (n=312) finish!

```
num: 0.0, x:3.0, n:312.0, p:0.012738853503184714
num: 1.0, x:10.0, n:312.0, p:0.03503184713375796
num: 2.0, x:9.0, n:312.0, p:0.03184713375796178
num: 3.0, x:10.0, n:312.0, p:0.03503184713375796
num: 4.0, x:10.0, n:312.0, p:0.03503184713375796
finalResult = 0.14655805977462028
```

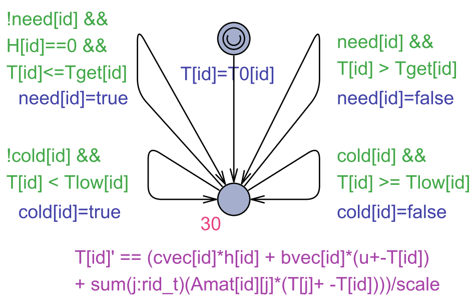
Property	Value
P=?[F energy>=2200]	

```
root-->f218-->n:250
1]-->f=3,n=3: [[0, 0, 1, 0, 0, 1]0], [skip0],
=29: [[0, 1, 1, 0, 0, 1]0], [[0, 0, 1, 0, 0, 1]0],
),n=29:
skip-->f=0,n=10:
[0, 1, 1, 0, 0, 1]-->f=0,n=
[0, 0, 1, 0, 0, 1]-->f=0,r
skip-->f=0,n=10:
[0, 0, 1, 0, 0, 1]-->f
[0, 1, 1, 0, 0, 1]--
```

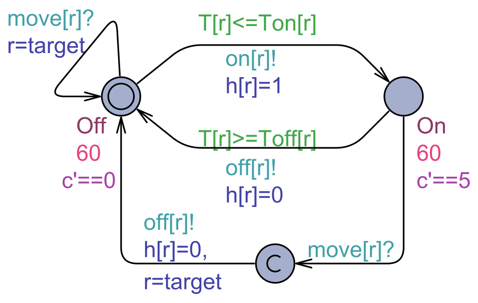
Parameters Setting

A Threshold :	0.5
ing Number :	250
ting Number :	4
Extracting Probability :	0.8
False Negatives :	0.05
Ratio Bound :	0.95

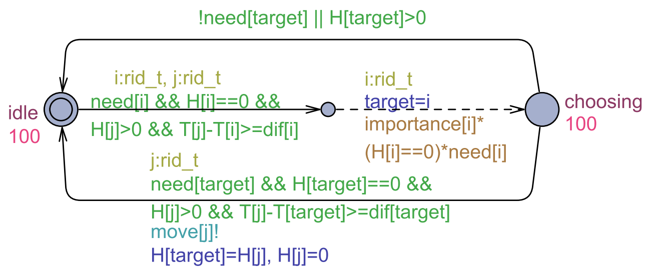
Figure 9. The user interface of Al-SMC verifier.



(a) Template for Room temperature



(b) Template for Heater control



(c) Template for Strategy

Figure 10. The main stochastic hybrid automata templates used in energy-aware building.

4.2 Evaluating energy-aware building with AL-SMC

UPPAAL-SMC[5] is a new version of UPPAAL which supports statistical model checking and adopts BIE algorithm. We compare the UPPAAL-SMC verifier with our AL-SMC verifier with the model of energy-aware building[7] which we have implemented in UPPAAL-SMC Model Checker. The goal of this model is to evaluate the comfort and energy consumption of various control strategies with varying environmental settings. The complete model consists of five SHA templates in parallel: rooms, heaters, central controller, weather, and a user profile for each room. Figure 10 shows main SHA templates in our model, in which each heater heats more than one room, and the room needs to be heated when the temperature is lower than a threshold. The controller decides how to move the heaters from one room to another. If one heater is needed by more than one room at the same time, the controller will choose a certain room according to the importance of each room. In this experiment, we use three properties to compare the efficiency and accuracy of UPPAAL-SMC verifier with AL-SMC verifier. The properties are as follows:

Table 3 Verified properties.

PID	Property
ϕ_1	$P_{=?}(F^{\leqslant 48}\ energy \geqslant 210)$
ϕ_2	$P_{=?}(F^{\leqslant 48}\ discomfort \geqslant 15)$
ϕ_3	$P_{=?}(F^{\leqslant 48}\ discomfort \leqslant 15 \wedge energy \geqslant 170)$

Table 4 Comparing traces number of AL-SMC with UPPAAL-SMC.

PID & Params (δ, c)	Min trace number (AL-SMC/ UPPAAL-SMC)	Max trace number (AL-SMC/ UPPAAL-SMC)	Mean trace number (AL-SMC/ UPPAAL-SMC)
ϕ_1(0.05,0.99)	250/578	353/647	350/625
ϕ_1(0.02,0.9)	842/1546	965/1644	959/1603
ϕ_2(0.05,0.99)	350/645	367/659	353/655
ϕ_2(0.02,0.9)	927/1677	935/1690	929/1685
ϕ_3(0.05,0.99)	363/662	412/684	387/672
ϕ_3(0.02,0.9)	861/1722	954/1810	882/1765

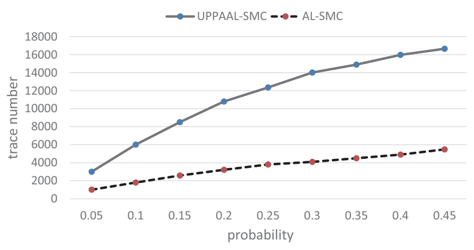
Table 5 Comparing time consumption of AL-SMC with UPPAAL-SMC.

PID & Params (δ, c)	Min time (AL-SMC/ UPPAAL-SMC)	Max time (AL-SMC/ UPPAAL-SMC)	Mean time (AL-SMC/ UPPAAL-SMC)
ϕ_1(0.05,0.99)	144/144	123/161	131/146
ϕ_1(0.02,0.9)	293/375	337/411	314/401
ϕ_2(0.05,0.99)	112/161	137/164	122/162
ϕ_2(0.02,0.9)	304/421	522/623	493/593
ϕ_3(0.05,0.99)	115/165	135/171	124/168
ϕ_3(0.02,0.9)	308/430	509/626	315/442

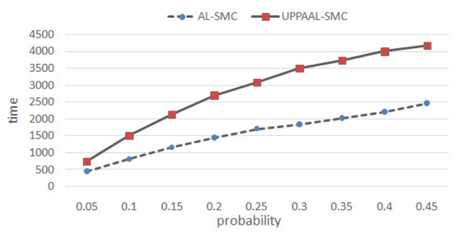
Table 6 Comparing probability of AL-SMC with UPPAAL-SMC.

PID & Params (δ, c)	Min probability (AL-SMC/ UPPAAL-SMC)	Max probability (AL-SMC/ UPPAAL-SMC)	Mean probability (AL-SMC/ UPPAAL-SMC)
ϕ_1(0.05,0.99)	0.33402/0.32759	0.48486/0.43914	0.41046/0.39084
ϕ_1(0.02,0.9)	0.34755/0.3553	0.46737/0.41981	0.40536/0.3909
ϕ_2(0.05,0.99)	0.42612/0.42813	0.56874/0.54116	0.50363/0.48225
ϕ_2(0.02,0.9)	0.44511/0.46099	0.54631/0.51391	0.49826/0.48473
ϕ_3(0.05,0.99)	0.44631/0.44823	0.53623/0.53821	0.48941/0.49633
ϕ_3(0.02,0.9)	0.45322/0.46721	0.53136/0.54817	0.48063/0.49312

We execute each verifier many times, and compare these two verifiers with three aspects: required trace number, time consumption and probability, respectively. Table 4 shows the number of required traces of two verifiers. It shows that AL-SMC verifier reduces the number of required traces more effectively, and the number of traces is reduced about 20% to 50% compared to UPPAAL-SMC verifier. Table 5 shows the comparison of time consumption of these two verifiers. It shows that AL-SMC verifier also reduces the time consumption, but the reduction is less obvious than that of the number of traces. The main reason is that AL-SMC simulates more variables for each state than UPPAAL-SMC does, so it consumes more time to simulate one trace. Table 6 shows the comparison of probability. In order to analysis the error between AL-SMC and UPPAAL-SMC, we estimate the real probabilty with a great many experiments, threrefore, the error of AL-SMC is the difference between the probabilty of AL-SMC and real probabilty, and the error of UPPAAL-SMC is obtained in the same way. We find that the probability error between UPPAAL-SMC and AL-SMC is acceptable within the error bound δ. To further estimate the error, we give the probability distribution of probability error for both UPPAAL-SMC and AL-SMC verifier which is shown in Fig. 12(a). The distribution of error approximates to the Student-T distribution[11]. To show the difference between UPPAAL-SMC verifier and AL-SMC verifier more intuitively, we test them under different probabilities and then record the time consumption and required traces as shown in Fig. 11. As we can see, the efficiency of AL-SMC verifier is enhanced obviously compared to UPPAAL-SMC verifier.

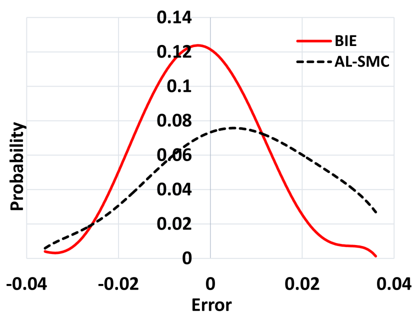


(a) Number of traces

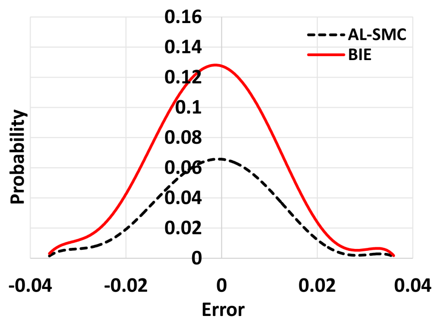


(b) Time consumption

Figure 11. Comparing trace number and time consumption of UPPAAL-SMC with AL-SMC.



(a) $r = 0.15$.



(b) $r = 0.25$.

Figure 12. Error estimation with different reduction degrees (r).

It is obvious that the less the number of simulated traces is, the greater the probability error is. It may be problematic when the number of traces for SMC is reduced too much, e.g. the case of $\phi_3(0.02, 0.9)$ whose number of simulated traces is reduced by nearly 50%. The reduction rate is actually determined by the number of end nodes in PFT. To decrease the number of end nodes can help increase that of simulated traces. As a result, we increase the parameter r to decrease the number of end nodes. As shown in Fig. 12(b), the error distribution can be effectively corrected by increasing r. By this way, we can balance the number of simulated traces (representing the efficiency of AL-SMC) and the error bound (representing the accuracy of AL-SMC) with r.

5 Related Works

We focus on the statistical model checking techniques which was first proposed by R.Grosu[12]. Based on the basic SMC, some variations have been proposed the past years[23,29,30,19,15,31]. Some related work are summarized as follows:

Performance of basic SMC. Kim et al.[20] give an empirical evaluation of the above algorithms except APMC, and the experimental result turns out that the four algorithms are practically useful to safety critical hybrid systems. For qualitative SMC, SSP is less effective than SPRT and BHT; BHT is faster than SPRT when checking the property whose the probability threshold is far away from its real probability, otherwise BHT is obviously less efficient than SPRT. Zuliani et al.[31] compare the number of traces analyzed by APMC and BIE; they conclude that BIE excels remarkably in performance. Our approach uses BIE algorithm to accomplish the probability estimation in the final phase.

Learning-based SMC. Our approach uses abstraction and learning techniques. Therefore we discuss the significantly related previous works which can be divided into three categories: (i) learning a prediction model for verification by means of predicting the possible result or occurrence of event[9,21]. This kind of methods can directly apply existing Machine Learning techniques, but hardly get a precise evaluation of error. (ii) learning probability distribution functions by statistical abstraction to simplify the original model without changing the probability distribution in the overall view[2]. It can give us an insight into the core behavior of a system, and may improve the performance as well as accuracy. (iii) learning probabilistic automata to build an abstract model with fewer states and transitions. Ref. [28] presented a wavelet-based approach to learning hybrid automata from a black-box system. Ref. [25] is aimed to improve the performance of SMC via verifying the learned abstract probabilistic automata (based on ALERGIA[24] as an extended grammar inference algorithm ALERGIA[6]). Ref. [26] abstracted the original probabilistic automata by invariant inference and effectively improved the SMC performance with an acceptable error. In fact, our approach can be treated as a combination of the 2nd and 3rd categories.

Advanced Topics of SMC. To further explore SMC technology, some advanced topics have been studied. Numerical and statistical methods are combined to improve the performance of SMC or to address the nuts like non-determinism[4,26]. Henriques et al. presented an approximate approach to non-determinism issue of SMC in a probabilistic way[14]. SMC is usually good at checking a time-bounded property, so how to check an unbounded property remains a hot topic. He, Jennings et al. presented a method for transforming an unbounded “Until” issue to bounded one[13,18]. Another topic of SMC is checking the property containing rare event because the extremely low occurrence of rare event always requires a large number of simulations, leading to unacceptable low efficiency. Jegourel, Legay et al. presented improved approach oriented to rare event based on the efficient simulation techniques (Importance Sampling and Importance Splitting)[16,17], which effectively reduce the number of traces required by SMC.

6 Conclusion

In this paper, we focus on improving the performance of SMC with automatic abstraction and learning techniques. SMC may encounter the performance bottleneck when it evaluates probability using BIE algorithm in some cases. To solve the problem, we propose an optimized approach for SMC based on abstraction and learning called AL-SMC. The novelty of our approach are: (i) simplifying the traces by projection and abstraction techniques; (ii) partitioning the original probability space into several relatively balanced sub-spaces by learning a PFT; (iii) evaluating the probability of each sub-space in parallel. Besides, we analyse the time and space complexity of some core algorithms, and have also implemented AL-SMC in our Modana Platform to support the automatic abstraction and learning. Experimental results show that the optimized AL-SMC can effectively improve the performance of SMC and ensure the accuracy of the result with an acceptable range.

As part of future work, in addition to focusing on the efficiency of our implementation, we plan to use a heuristic method to optimize AL-SMC by automatically adjusting the parameter setting. Furthermore, we will improve our tool to make it more user-friendly.

Acknowledgements

This work was supported by NSFC (Grant No.61472140, 61202104) and NSF of Shanghai (Grant No. 14ZR1412500).

References

[1] Baier C, Katoen JP, et al. Principles of model checking. MIT Press Cambridge. 2008. 26202649.

[2] Basu A, Bensalem S, Bozga M, Caillaud B, Delahaye B, Legay A. Statistical abstraction and model-checking of large heterogeneous systems. Formal Techniques for Distributed Systems, Springer, 2010: 32–46.

[3] Cheng B, Wang X. Modana: An integrated framework for modeling and analysis of energy-aware cpss. Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual. IEEE. 2015, 2. 127–136.

[4] Bogdoll J, Fioriti LMF, Hartmanns A, Hermanns H. Partial order methods for statistical model checking and simulation. Formal Techniques for Distributed Systems. Springer Berlin Heidelberg, 2011: 59–74.

[5] Bulychev P, David A, Larsen KG, Mikučionis M, Poulsen DB, Legay A, Wang Z. Uppaal-smc: Statistical model checking for priced timed automata. arXiv preprint arXiv: 1207.1272. 2012.

[6] Carrasco R C, Oncina J. Learning stochastic regular grammars by means of a state merging method. International Colloquium on Grammatical Inference. Springer Berlin Heidelberg. 1994. 139–152.

[7] David A, Du D, Larsen KG, Mikučionis M, Skou A. An evaluation framework for energy aware buildings using statistical model checking. Science China information sciences, 2012, 55(12): 2694–2707.

[8] David A, Larsen KG, Legay A, Mikučionis M, Poulsen DB, Van Vliet J, Wang Z. Statistical model checking for networks of priced timed automata. Formal Modeling and Analysis of Timed Systems. Springer. 2011. 80–96.

[9] Du DH, Cheng B. Statistical model checking for rare-event in safety-critical system. Journal of Software, 2015, 26(2): 305–320.

[10] Dunteman GH. Principal Components Analysis. Sage, 1989, 69.

[11] George EO, El-Saidi M, Singh K. A generalized logistic approximation of the student t distribution. Communication in Statistics-Simulation and Computation, 1986, 15(4):

1199–1208.

[12] Grosu R, Smolka SA. Monte carlo model checking. International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer. 2005. 271–286.

[13] He R, Jennings P, Basu S, Ghosh AP, Wu H. A bounded statistical approach for model checking of unbounded until properties. Proc. of the IEEE/ACM international Conference on Automated Software Engineering. ACM. 2010. 225–234.

[14] Henriques D. Statistical model checking for markov decision processes[PhD Thesis]. General Motors, 2012.

[15] Hérault T, Lassaigne R, Magniette F, Peyronnet S. Approximate probabilistic model checking. Verification, Model Checking, and Abstract Interpretation. Springer. 2004. 73–84.

[16] Jegourel C, Legay A, Sedwards S. Cross-entropy optimisation of importance sampling parameters for statistical model checking. Computer Aided Verification. Springer. 2012. 327–342.

[17] Jegourel C, Legay A, Sedwards S. Importance splitting for statistical model checking rare properties. Computer Aided Verification. Springer. 2013. 576–591.

[18] Jennings P, Ghosh AP, Basu S. A two-phase approximation for model checking probabilistic unbounded until properties of probabilistic systems. ACM Trans. on Software Engineering and Methodology (TOSEM), 2012, 21(3): 18.

[19] Jha SK, Clarke EM, Langmead CJ, Legay A, Platzer A, Zuliani P. A bayesian approach to model checking biological systems. Computational Methods in Systems Biology. Springer. 2009. 218–234.

[20] Kim Y, Kim M, Kim TH. Statistical model checking for safety critical hybrid systems: An empirical evaluation. Hardware and Software: Verification and Testing. Springer. 2012. 162–177.

[21] Kumar JA, Ahmadyan SN, Vasudevan S. Efficient statistical model checking of hardware circuits with multiple failure regions. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2014, 33(6): 945–958.

[22] Lee EA. The past, present and future of cyber-physical systems: A focus on models. Sensors, 2015, 15(3): 4837–4869.

[23] Legay A, Delahaye B, Bensalem S. Statistical model checking: An overview. International Conference on Runtime Verification. Springer Berlin Heidelberg. 2010. 122-135.

[24] Mao H, Chen Y, Jaeger M, Nielsen TD, Larsen KG, Nielsen B. Learning probabilistic automata for model checking. 2011 Eighth International Conference on Quantitative Evaluation of Systems (QEST). IEEE. 2011. 111–120.

[25] Nouri A, Raman B, Bozga M, Legay A, Bensalem S. Faster statistical model checking by means of abstraction and learning. International Conference on Runtime Verification. Springer. 2014. 340–355.

[26] Pavese E, Braberman V, Uchitel S. Automated reliability estimation over partial systematic explorations. 2013 35th International Conference on Software Engineering (ICSE). IEEE. 2013. 602–611.

[27] Pearson K. On lines and planes of closest fit to system of points in space. Philiosophical Magazine, 1901, 2(6): 559–572.

[28] Vodenčarević A. Learning behavior models of hybrid systems using wavelets for autonomous jumps detection. 2012 10th IEEE International Conference on Industrial Informatics (INDIN). IEEE. 2012. 151–156.

[29] Younes HL. Verification and planning for stochastic processes with asynchronous events [Technical Report]. DTIC Document. 2005.

[30] Younes HL, Simmons RG. Statistical probabilistic model checking with a focus on time-bounded properties. Information and Computation, 2006, 204(9): 1368–1409.

[31] Zuliani P, Platzer A, Clarke EM. Bayesian statistical model checking with application to stateflow/simulink verification. Formal Methods in System Design, 2013, 43(2): 338–367.