A Method of Predicting Software Behavior Risk based on Off-line Runtime Verification

Lei Hu[1, a], Guohua Jiang[2, b]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[2] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[a]email: 919074855@163.com, [b]email: jianggh@nuaa.edu.cn

Keywords: software behavior risk; runtime verification; prediction

Abstract. The current methods of software behavior risk prediction is mainly through the study of the operating rules from the data of the other software of the same type, and that leads to differences between the prediction results and the actual software behavior. Aiming at this problem, this paper presents a software behavior prediction method, which combines prediction of software behavior with runtime verification, using Markov Chain and Hidden Markov Model(HMM), to analys the data from offline runtime verification and predict software behavior. Experiments show that this method can significantly improve the accuracy of the prediction.

Introduction

Usually, two of the main meanings of software behavior prediction are: the prediction of users' interaction behavior, for prevention of fraud and other abnormal behavior from user[1], and the prediction of the quality of software by establishing a software reliability model [2]. The method mentioned in this paper was neither of them. Here, the software behavior prediction refers to predicting the future paths of software.

Runtime validation is different from the formal verification of model checking methods.In runtime verification, according to different ways to work, it can be divided into offline verification and online verification [3].In online validation, it's mainly by monitoring the process of real-time monitoring, to achieve found abnormal operation timely. In offline verification mode, it is mainly about the analysis of the operation of history, or through the process of differentiation, the monitoring process is independent of the target system. General online run-time monitoring process is as fellow: (1). insert the pile in target program, the way of pile inserted has manual and automatic [4] two kinds, Its aim is pass the target system code execution and variable values such as running condition to the event recognizer (2).According to the nature of the protocol to be verified, give the definition of events that need to be concerned. The definition of an event is usually given by human, because the nature of the protocol to be verified is usually also artificially selected.(3). Target system operation turn generate execution sequences of events related to property specification process is called event recognition, through an event recognition and monitor the nature of protocol verification module to the software running status sequence of analysis and verification results are given.

The software behavior prediction method proposed in this paper directly uses the the future path data from target software itself to predict, and can more accurately respond to the operation regular of the software. By collecting and statisting the data of monitoring software in the real environment, and build probabilistic models with Markov Chain and HMM[5][6], the method can give a probabilistic model of the possible paths in the future.

The collection of software behavior information and the establishment of probability model

To collect the information of software behavior, software information which is collected by users need to be defined. The definitions is described by PEDL and MEDL language[]. In this way, instrumentation on the target software will be done.And from that, a state sequence{X(n)} can be obtained. By calculating the number of state j transfer to state K in the sequence {X (n)} we can get the transfer frequency F_{jk}, and then construct the one step transfer matrix of the sequence of {X (n)} as follows:

$$F = \begin{bmatrix} F_{11} & \cdots & \cdots & F_{1m} \\ F_{21} & \cdots & \cdots & F_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ F_{m1} & \cdots & \cdots & F_{mm} \end{bmatrix} \tag{1}$$

We can get a one-step transtion-probablity matrix P_{jk}, in which:

$$P_{jk} = \begin{cases} \dfrac{F_{jk}}{\sum\limits_{j=1}^{m} F_{jk}}, \sum\limits_{j=1}^{m} F_{jk} > 0, \\ 0, \sum\limits_{j=1}^{m} F_{jk} = 0, \end{cases} \tag{2}$$

The above is to establish the Markov chain model for the software behavior probability of the system which does not contain the hidden state. To the system that includes the hidden states hidden Markov chain model, the problem that adjusting the model parameters belongs to the problem learning hidden Markov model, EM[7] algorithm can solve this problem.

Method of software behavior prediction

In the method proposed in this paper, how to forecast software behavior is the main problem, this section will describe two methods: Basic method of software behavior prediction and Extend method of software behavior prediction. The first method is aimed at the systems do not include the hidden state and the second is aimed at the systems contain some hidden state.

Basic method of software behavior prediction

The basic method of software behavior prediction is based on Markov chain to predict the behavior of the software. Markov chain is used to describe the discrete event with characteristics of markoff random process. It describes a state sequence, the value of each state depends on the front of a finite state, if it is only related to the previous state, Is called first-order Markov chain.For first-order Markov chain, there is the following theorem:

in the sequence, the formation type:

$$\begin{aligned} &P(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n) \\ &= P(X_{n+1} = x \mid X_n = x_n) \end{aligned} \tag{3}$$

Collection {X_1, X_2, X_3…} is called "state space". The changes of the relationship between state is represented by the state transition probability matrix.

In this section,Markov chain state spaces is a state collection of software system, state transition matrix describes the transition relationship between the system states in a system running process. Collect information about software behaviors to obtain the state transition matrix. Due to the limitation of the markov chain, it can only calculate the probability of reaching a particular state through specific steps, and does not directly show the appearance probability of a particular path sequence. As a result, the problem is converted into obtaining all possible sequences of state within the next N steps in a certain state, and calculate the appearance possibility of each sequence,then calculate operational risks in the future.

The following steps are given to calculate the risk of future operations:

(1) Collect information about software running to obtain $S=\{S_1,S_2,\ldots S_n\}$ (the collection of the software running states) and $P(1)$ (the state transition matrix in one step).

(2)Set the steps need to predict as N, according to the state transition matrix, if the current state is $S_i(1\leqslant i\leqslant n)$, the states which state transition probability is non-zero are the reachable states, gain the list of the next reachable states and mark the transition probability.

(3)Build a tree, set S_i as the root, set the reachable states of S_i as it's child nodes, then put each child node as a new parent node, set the reachable states of it as it's child nodes. Until the depth of the tree as N+1. Here attached to each child node transition probability are one step transition probability, is the transition probability from it's parent node to it.

(4)In the tree, each leaf node in layer N+1 represents a possible software execution path. In all paths from the root node to leaf node, enumerate all possible path meet the property specification, according to the transition probability marked in the node, calculate the appearance possibility of each path, call it $P_{Si.}$ Here is the formula to calcalute P_{Si}:

$$P=P\left(\ln_j\right)\bullet P\left(F(\ln_j)\right)\cdots P\left(F^{N-2}\left(\ln_j\right)\right) \tag{4}$$

(5)Sum all possible path which meet the property specification P_{Si} obtain ΣP_{si}, in current state, the risk value of software execution in N next steps is $1\text{-}P_{si}$.

(6) Repeat the above steps, set the states in state set S as the current states, after calculating the execution risk within finite steps in the future, stores the result in data structures. When need these data can be directly query retrieval.

Extend method of software behavior prediction

For most of the software system, if we want to perform runtime validation, we can get system operation information through some monitoring means, for example program inserting pile. But some software may exist this situation: During the monitoring, the monitoring information cannot be directly on behalf of the change of the system state, and it only represents a observable event occurred in the current state of the system. We can't intuitively observe the change of system state. As shown in Fig.1, $<s_1,s_2,s_3>$ represents an unobservable system state sequence, and $<o_1,o_2,o_3>$ represents an observable event sequence. At the runtime validation, we can only obtain an observable event sequence, the system state transition probability and the happening probability of an observable event in a certain system state. At this situation, we introduce the HMM to solve the software behavior prediction problem.

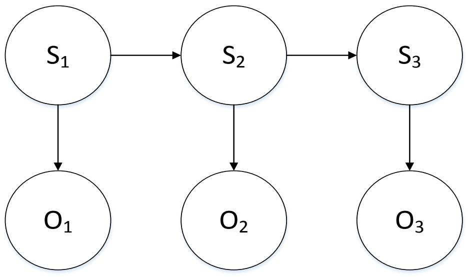


Fig.2. System containing hidden state

HMM is used to describe a Markov process containing implicit unknown parameter. A HMM can be represented by a quintuple: $H=(S,A,V,B,\pi)$, and S represents a set of implicit state, and A represents a matrix of implicit state transition probability, and V represents a set of observable states, and B represents a matrix of observable state transition probability(also called as Confusion Matrix), and π represents a probability matrix of initial state. Usually we can use a triple $H=(A,B,\pi)$, to concisely represent a Hidden Markov Model. Among them, the elements of S and V can be enumerated. They can be respectively expressed as $S=\{s_1,s_2,\ldots s_{Ns}\}$and $V=\{v1,v2,\ldots,v_{No}\}$. N_s

represents the number of implicit states, and N_o represents the number of observable states. A is a $N_s \times N_s$ matrix, and $A_{ij}=P(s_j|s_i),1\leqslant i,j\leqslant N$. B is a $N_s \times N_o$ matrix, and $B_{ij}=P(v_j|v_i),1\leqslant i,j\leqslant N_o$. π is the probability that the initial state status is s_i. HMM mainly can be used to solve three categories of problems : assessment problems, decoding problems and learning problems. But in this section, the problem to be solved is that under the condition of the current observable state and HMM, calculate the risk of subsequent possibly happened observable state sequence. This is not the same as the above three kinds of problems, so this section gives the concrete steps to solve the problem as follows:

(1)Based on HMM and the state s_i, build the tree of future state sequence in N step(the same as the steps (1), (2), (3) in the section).

(2)Repeat the step (1), build the same tree for all states in the set S of implicit states.

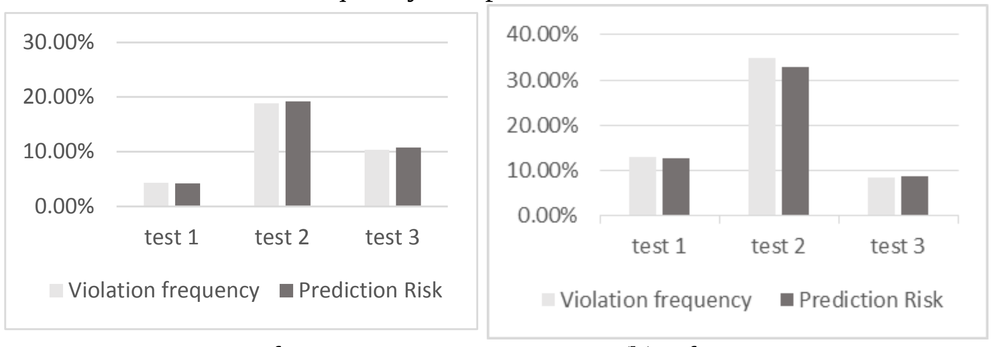
(3)If the current observable state is v_i, enumerate all the implied states possibly leading to v_i. And list possible paths (from the root node to leaf node) that meeting the property in the trees. According to the marked transition probability, calculate the happening probability of each path p_{si}.

(4)After getting the corresponding implicit state path sequence, get all the observable state sequences which may be corresponding with each implicit state sequence. Determine them whether meet the property and sum the probability Σp_{si}. The risk of software running in N steps is $1-\Sigma p_{si}$.

(5)Repeat the above steps, calculate their running risk in future N steps for all the observable states , and store them in a data structure.

Experiment result

Apply the methods to two software systems, basic method of software behavior prediction for software A, and extend method of software behavior prediction for software B. Collect their historical data, and calculate the future operational risk. Run A and B for enough times, get the frequency of violating the specification. Verify that the calculated run risk is consistent with the actual results or not. The violation frequency and prediction risk of A and B is shown in Fig.2.



(a) software A (b)software B

Fig.2.Violation frequency and Prediction Risk

From the data in Fig.2, we can see that the violation frequency and the prediction risk are very close, which means that the methods in this.paper can accurately predict the risk of software operation.

Conclusion

Based on the analysis of the mechanical theory as the foundation, designed the soccer robot pick the ball institutions optimal design process, found aim function, select design variables and the corresponding optimization algorithm to optimize a complete set of institutions. At last through the test to get the final performance parameters of the institution. Experiments show that the system has higher accuracy and stability, the new optimize pick the ball have design basic requirements, and achieved good ideal control effect.

Based on the On collecting the data of the software behavior, this paper designed methods to predict the future behavior and risk of the software based on the historical data, and puts forward the basic method of software behavior prediction and the extend method of software behavior prediction. Finally, the predicted results are compared with the actual situation in the experiment. The results show that the method proposed in this paper has high accuracy and can effectively predict the future operational risk of the software.

Acknowledgement

In this paper, the research was sponsored by Software Testing Center of College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics.

References

[1] Wang Q, Yu B, Zhu J. Extract Rules from Software Quality Prediction Model Based on Neural Network[C]// null. IEEE Computer Society, 2004:191-195.

[2] Lee K C, Ozdemir H T, Yu J. System and method for predicting abnormal behavior: US, US 20100207762 A1[P]. 2010.

[3] Pike L, Goodloe A, Morisset R, et al. Copilot: a hard real-time runtime monitor[C]//Runtime Verification. Springer Berlin Heidelberg, 2010: 345-359.

[4] Ye hua L I, Nai jie G U, Zhang Y N, et al. Runtime Program Verification Framework Based on Instrumentation and Boolean Logic[J]. Computer Engineering, 2013, 39(1):29-17.

[5] Keilson J. Markov chain models—rarity and exponentiality [M]. Springer Science & Business Media, 2012.

[6] Trivedi K S, Vaidyanathan K, Selvamuthu D. Markov chain models and applications [C] // Elsevier Inc. 2015.

[7] Yildirim S, Singh S S, Doucet A. An online expectation–maximization algorithm for changepoint models [J]. Journal of Computational and Graphical Statistics, 2013, 22(4): 906-926.