Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 8, August 2018, pp. 1796-1803
DOI: 10.23940/ijpe.18.08.p16.17961803

Formal Analysis and Verification of Timing and Resource Adaptability for Internetware

Zhongqun Wang, Jun Li*, and Qi Xia

School of Computer and Information, Anhui Polytechnic University, Wuhu, 241000, China

Abstract

To address the trustworthiness of Internetware under open and dynamic environments, this paper proposes an approach to verify whether Internetware is satisfied with timing and resource constraints. Firstly, interface automata are extended with time and resource semantics. Then, timing and resource interface automata is used to model the behaviors of a software component. An algorithm is also provided to check whether all behaviors of an Internetware system are satisfied with resource constraints within a specified time. Finally, an online bookstore system is employed to illustrate our work, and the model checker Spin is used to verify the correctness of our approach.

Keywords: internetware; interface automata; timing and resource constraints

(Submitted on May 11, 2018; Revised on June 16, 2018; Accepted on July 17, 2018)

1. Introduction

Internetware are usually run in an environment with timing and resource constraints [1-3]. Effectively analyzing and verifying the behavior changes of component compositional systems, which are caused by environmental changes, and guarantee the trustworthiness of systems is very challenging. Previous works [4-5] extended interface automata by considering timing and resource constraints in terms of timed interfaces and resource interfaces, which are used to formally analyze and verify component behaviors under real-time or resource-constrained environments, respectively. Hu et. al [6] proposed a method that uses interface automata with resource utilization information to formally model and analyze the behaviors of components. In [7], the reliability transition matrix of an Internetware system was constructed using Internetware system architecture and considering the accumulation effects of Internetware reliability changes. In [8], a trust model for Internetware based on a complex network was proposed, and the characteristics of the small-world and scale-free complex network was introduced into this model. Trust computation rules and proof for its dynamic evolution were given. In [9], the testing requirements and testing steps of internetware comprehensive testing were described, and a specific method for internetware comprehensive testing was put forward from the perspectives of application components testing. In order to save development time, cost, and keep original features, Vitrand et al. [10] proposed an approach to support existing architecture evolution in Internetware. That is, the approach includes three phases: 1) prediction 2) transformation 3) comparison, which not only makes the systems more self-adaptive in Internetware environments, but can also guide system evolution.

In our previous works [11-12] we presented a method to improve the trustworthiness for Internetware evolution through verifying business consistency and satisfaction of random resources. But, our previous works were based on separately looking at timing constraint and resource constraint and did not consider that computing environments are limited by timing and resource constraints at the same time. Real-world applications, such as online bookstores and e-mail systems, are often unavailable or timed-out because of a lack of bandwidth resources or high bandwidth delay. Therefore, the investigation of Internetware trustworthiness should take timing and resource constraints into account.

* Corresponding author.
E-mail address: edmondlee@ahpu.edu.cn

By taking timing limitation into consideration, this paper first extends resource interface automata and proposes a timing and resource interface automata, which is used to describe the combined behavior of the Internetware components with timing and resource constraints. Then, this paper studies how to model Internetware components constrained by time and resources using timing and resource interface automata. An algorithm checks whether components satisfy resource requirements within a pre-specified time. An online bookstore system is offered as a case study. Finally, the model checker Spin is used to validate the correction of our model.

The paper is organized as follows. Section 2 extends the interface automata using timing and resource semantics. Section 3 gives an algorithm to check whether a component compositional system is satisfied with timing and resource constraints. A case study is conducted in Section 4. In Section 5, we verify our model using the model checker Spin. In the last section, we draw conclusions and give our future work for this research.

2. Timing and Resource Interface Automata

Resource interface automata [6] are used to describe the behaviors of software component resource utilization. We extended resource interface automata with a timing constraint and the extended automata is called the timing and resource interface automata. For the sake of convenience, we first made the following assumptions about resource and time:

(1) Resource is non-shared. One resource can only be utilized by one component. No other components can use this resource simultaneously.

(2) Both resource and time can be quantified. They are expressed as integer multiple of the minimum unit. This paper will take network bandwidth as a resource example and quantify it. The unit of the network bandwidth will be 256 (KB/S), and the unit of time will be 1 (ms).

Timing and resource interface automata are used to model the online bookstore system [11]. Figure 1 is a resource interface automata model for the seller in the online bookstore system with extended timing semantic. It describes demands of the system's network bandwidth resource with a timing constraint, where every state is marked with an ordered number, where x, y, and k are non-negative integer and $x<y$. The ordered number indicates time demand on the network bandwidth resource: k units of the network bandwidth resource in (x, y) time bound must be available. At the initial state 0: resource utilization information is <(1,2), 2>, and this means that the component at state 0 must take two units of network bandwidth resource within 1~2 (ms). If two units of the network bandwidth resource are not available forever, the system will not function normally. If two units of network bandwidth resource could be provided but are not available within 1~2 (ms), the system will issue a message timeout. The resource utilization of the seller will be obtained from the above analysis about state 0 in Figure 1. By analyzing resource utilization for each state in this way, we are able to know if the system can work well under a computational environment with a fluctuation of available resources.

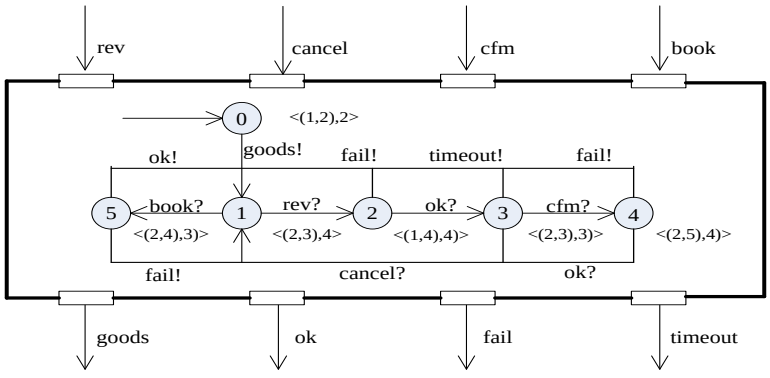


Figure 1. Role seller timing resource interface automata

2.1. The Formal Definition of Timing and Resource Interface Automata

Formal definition for timing and resource interface automata is as follows:

Definition 1 a timing and resource interface automation is defined as $P =< V_P, V_P^{init}, A_P^I, A_P^O, A_P^H, F_P, W_P, \Delta_P >$, and

- V_P is a set of states.
- $V_P^{init} \subseteq V_P$, is a set of initial states, which contain one state at most.
- A_P^I, A_P^O, A_P^H are mutually disjoint sets of actions. They are called input actions, output actions, and internal actions. The set of all actions are denoted as A_P, and $A_P = A_P^I \cup A_P^O \cup A_P^H$.
- F_P is a set of mappings, $F_P =< (x, y), k >$ (x, y and k are non-negative integer, $y \neq \infty$ and $x < y$). x represents the lower bound of minimum time for a system to obtain resources, y stands for the upper bound of minimum time. $F_P(v_i)$ specifies whether resources in state v_i will meet the time bound.
- W_P is a set, and each element of W_P has the form: $w = \{(v, F_P(v_i)) | v_i \in V\}$.
- Δ_P is a transition relation, which represents a set of transition steps. $\Delta_P \subseteq W_P \times A_P \times W_P$. Each $w \subseteq W_P$ is a state with timing and resource constraints. $F_P(v_i)$ stands for time and resource limitation information about state v_i.

The behavior definitions in timing and resource interface automata are similar to those in resource interface automata [6]. The state transition sequence for a component denotes the information for resource utilization in the process of state transition for the component. In Figure 1, the state transition sequence for the seller is as follows:

- $\varphi = \{0, <(1,2), 2>\} \xrightarrow{goods} \{1, <(2,3), 4>\} \xrightarrow{rev?} \{2, <(1,4), 4>\} \xrightarrow{ok?} \{3, <(2,3), 3>\} \xrightarrow{timeout!} \{1, <(2,3), 4>\}$. The following is the timing limitation information sequence of resource utilization in each state derived from the above state transition sequence φ.
- $\lambda =< (1,2), 2 >^{\wedge} < (2,3), 4 >^{\wedge} < (1,4), 4 >^{\wedge} < (2,3), 3 >^{\wedge} < (2,3), 4 >$. Obviously, through λ, we can clearly observe an action's time information about resource utilization for the component.

2.2. Timing and Resource Interface Automata Network

Timing and resource interface automation network is used to describe behaviors of component compositional systems with timing and resource constraints for Internetware. The formal definition of the timing and resource interface automation network is similar to that of the resource interface automaton network, which is precisely defined in [6]. The following is the state set and action set of the timing and resource interface automation network.

Definition 2 Assume $N = (Q, Z)$, the combined state set and action set are defined as follows:

- Each combination in N has the form: $\bar{v} = (v_1, v_2, \cdots, v_n)$, and $v_i \in V_{P_i}$. Combined state set is $V_N = V_{P_1} \times V_{P_2} \times \cdots \times V_{P_n}$.
- Each combination that consists of timing and resource constraints in N has the form: $\bar{w} = (\bar{v}, F_N(\bar{v}))$ and $F_N(\bar{v})$ is the ordered set that stands for N's resource utilization characteristics with timing constraint: $< (x_N(\bar{v}), y_N(\bar{v})), k_N(\bar{v}) >$.
- The actions set of N is defined as $A_N = A_N^I \cup A_N^O \cup A_N^H$. Input action is $A_N^I = (\bigcup_{1 \le i \le n} A_{P_i}^I) / Z$. Output action is $A_N^O = (\bigcup_{1 \le i \le n} A_{P_i}^O) / Z$. Inner actions set is $A_N^H = (\bigcup_{1 \le i \le n} A_{P_i}^I) \cup Z$.

3. Check Whether Compositional System is Satisfied with Timing and Resource Constraints

The resources required by the component compositional systems runtime must be provided in an environment within a specified time. The resource ε information provided for components in the environment would be denoted by $L = (t_i, k_i)$, where t_i is the time when the resource is provided in the environment in state i, and k_i is the amount of resource units

provided in the environment at state i. Obviously, t_i and k_i are the timing and resource constraints for the component, respectively.

The literature [13] presented a compatible interface automation network and pointed out that the reached graph for state space constructed using Petri-net is equivalent to those of the interface automation network. Thus, reached graph constructed by timing and resource interface automata is of the state space of the interface automation network, and the resource information of reached graph is denoted by $X_{s_i} =< (x_{s_i}, y_{s_i}), k_{s_i} >$, where $x_{s_i} < y_{s_i}$. For a component, if $k_i > k_{s_i}$ and $x_{s_i} \le t_i \le y_{s_i}$ for any state, then the node is normal and the compositional system satisfies the resource constraint within the specified time bound; if $k_i \ge k_{s_i}$ and $t_i \notin (x_{s_i}, y_{s_i})$, then the system satisfies the resource constraint but does not satisfy the time limitation; if $k_i < k_{s_i}$ and $x_{s_i} \le t_i \le y_{s_i}$ for any state, then the system satisfies the time constraint but does not satisfy the resource limitation; if $k_i < k_{s_i}$ and $t_i \notin (x_{s_i}, y_{s_i})$ for any state, then the system does not satisfy resource or time limitations. The reached graph in literature [12] is extended to get the reached graph in Figure 2. If the resource provided for state S_7 for the compositional system in the environment is true for $k \ge 3$ and $3 \le t \le 5$, then the state S_7 is not constrained by the computing environment.

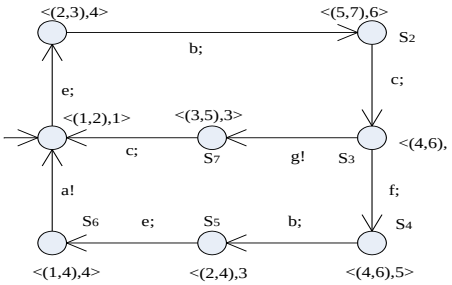


Figure 2. Reached graph

Based on the above discussion, Algorithm 1, which is the improvement of the algorithm in [12], is given to verify whether the component compositional system satisfies resource and timing constraints. The input values in Algorithm 1 are the reached graph and resource ε utilization information $L = (t_i, k_i)$, and the output values are whether the compositional system satisfies resource and timing constraints. In Algorithm 1, the traverse path is stored in the current path. Resource information about each state is in L, and $X(nodes)$ is denoted as the resource utilization information for the *node*, where x_i and y_i are the upper and lower bounds of time that required resources are allocated for state i respectively; X_i is the number of resource units needed for state i. *Nodes* where resource and timing constraints are not met are stored in variable *abnormal*. *Nodes* where timing constraints are met, but resource constraints are not met are stored in variable *realtime*; *Nodes* where resource constraints are met, but timing constraints are not met are stored in variable *resource*; variable *satisfied* is true if abnormal *nodes* exist, and is false otherwise.

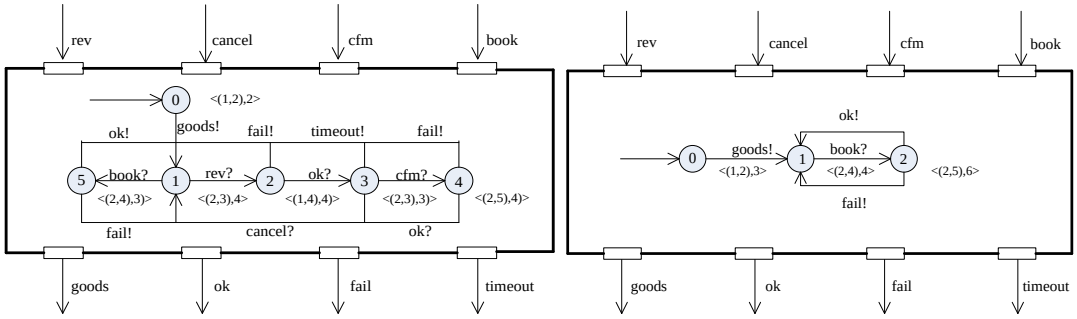
Algorithm 1 checks whether a compositional system is satisfied with resource and timing constraints within a specified time bound.

$current\_path := \{S_0\}; L =< t_i, k_i >;$
$X :=< (0,0), 0 >; abnormal := \varnothing; realtime := \varnothing; renormal := \varnothing; satified := true;$
repeat
$node :=$ the last node of $current\_path$
if *node* has no new successive node **then** delete the last node of $current\_path$;
else begin $node :=$ a new successive node of *node*;
$X := X(node)$;
for $i = 1$ to K
if $X_i > k_i \,\&\&(t_i < x_i \,||\, t_i > y_i)$ **then** $satisfied := false$;
{
if $satisfied := false$ **then** append node to *abnormal*;
append node to $current\_path$;

}
if $X_i \le k_i \&\&(t_i < x_i \,||\, t_i > y_i)$ **then** $satisfied := false$;
{
if $satisfied := false$ **then** append node to *resource*;
append node to $current\_path$;
}
if $X_i > k_i \&\&(x_i \le t_i \le y_i)$ **then** $satisfied := false$;
{
if $satisfied := false$ **then** append node to *realtime*;
append node to $current\_path$;
}
End
until $current\_path = \{\ \}$;
if $abnormal := \varnothing \&\& resource := \varnothing \&\& realtime := \varnothing$ **then return** ture;

4. A Case Study

An online bookstore system fails or responds to client timeout due to the fluctuation of network bandwidth. Thus, the online bookstore system would satisfy resource and timing constraints for network bandwidth. We use resource and timing interface automata to describe the utilization of network bandwidth for the online bookstore system. The resource and timing interface automata descriptions for the seller and interface of the component publisher for the online bookstore system after its evolution are shown in Figure 3(a) and 3(b), respectively.



(a) Role seller (b) Interface publish

Figure 3. Timing and resource interface automata for online bookstore system

Take state 2 in Figure 3(a) for example, when the network bandwidth provided in environment is 1(M/S) and the timeout is 3(ms) in state 2, that is, the network bandwidth provided in the environment is 4 units and the timeout is 3 units, the online bookstore system can satisfy resource and timing constraints and clients can use it properly. When the network bandwidth provided in the environment is 768(KB/S), and the timeout is 3(ms) in state 2, the system satisfies timing constraints but does not satisfy resource constraints. So, clients cannot use the online bookstore system properly. When the network bandwidth provided in the environment is 1(M/S), and the timeout is 5(ms) in state 2, the system is not limited by the resource constraint but does not satisfy the timing constraint, and thus, the system would timeout.

5. Our Model Verification

To verify the role seller model in the online bookstore system after evolution, the fact that resource constraints in state 1 and timing constraints in state 2 are met was validated by the model checker Spin in the literature [13-14]. According to transformation rules from the interface automation model [15-16], the model in Figure 4 is transformed into the Promela model as below. In the Promela model variables k and t are the number of network bandwidth units provided and the time when resources are available in state 1, respectively. Variables j and v are the number of network bandwidth units provided and the time when resources are available in state 2, respectively. Due to the limitations of space, parts of the codes for the Promela model are given as follows.

```
mtype={good,rev,ok,fail,cfm,cancel,Timeout}
proctype Publisher(chan ch1,ch2,ch3,ch4,ch5,ch6)
{ int k=,j=,t=,v=;
  do::
  sv0:
  do::
    if
     ::(t>=2&&t<=5)&&(k>=4)->ch!goods;
goto sv1;
fi
od;
  sv1:
  do::
  if
     :: (j>=4)&&(v>=1&&v<=4)->ch2?rev;
goto sv2;
  fi;
od;
sv2:
  if
      ::ch3!fail->goto sv1;
      ::ch3!ok->goto sv3;
    fi;
 sv3:
    if
     ::ch5?cancel->goto sv1;
     ::ch5?cfm->goto sv4;
     ::ch5!timeout->goto sv1;
    fi;
 sv4:
    if
      ::ch6!ok->goto sv1;
      ::ch6!fail->goto sv1;
    fi;
 od
}
```

For the above model described by Promela, we verify that the state space of the system is reachable under conditions of $< k = 4, t = 3, j = 4, v = 2 >$, $< k = 3, t = 3, j = 4, v = 2 >$ and $< k = 4, t = 3, j = 4, v = 5 >$ by using the model checker Spin. Obviously, for the first case, the verification conditions are that resource and timing constraints are met in states 1 and 2. For the second case, the verification conditions are that resource constraints are met, but timing constraints are met in state 1, whereas resource and timing constraints are met in state 2. For the third case, the verification conditions are that resource and timing constraints are met in state 1, and timing constraints are not met, but resource constraints are met in state 2. Our verification results are as below.

```
State-vector 100 byte, depth reached 34, errors: 0
      160 states, stored
       84 states, matched
      244 transitions (= stored+matched)
        1 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.018       equivalent memory usage for states (stored*(State-vector + overhead))
    0.286       actual memory usage for states (unsuccessful compression: 1613.21%)
                state-vector as stored = 1855 byte + 16 byte overhead
    2.000       memory used for hash table (-w19)
    0.343       memory used for DFS stack (-m10000)
    2.539       total actual memory usage
```

(a) The verification result for $< k = 4, t = 3, j = 4, v = 2 >$

```
State-vector 100 byte, depth reached 2, errors: 0
        2 states, stored
        0 states, matched
        2 transitions (= stored+matched)
        1 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.000       equivalent memory usage for states (stored*(State-vector + overhead))
    0.286       actual memory usage for states (unsuccessful compression: 129056.90%)
                state-vector as stored = 149690 byte + 16 byte overhead
    2.000       memory used for hash table (-w19)
    0.343       memory used for DFS stack (-m10000)
    2.539       total actual memory usage
```

(b) The verification result for $< k = 3, t = 3, j = 4, v = 2 >$

```
State-vector 100 byte, depth reached 6, errors: 0
        6 states, stored
        0 states, matched
        6 transitions (= stored+matched)
        1 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.001       equivalent memory usage for states (stored*(State-vector + overhead))
    0.286       actual memory usage for states (unsuccessful compression: 43018.97%)
                state-vector as stored = 49886 byte + 16 byte overhead
    2.000       memory used for hash table (-w19)
    0.343       memory used for DFS stack (-m10000)
    2.539       total actual memory usage
```

(c) The verification result for $< k = 4, t = 3, j = 4, v = 5 >$

Figure 4. The verification results using model checker Spin

6. Conclusions

To address the level of trustworthiness for Internetware's adaptation under an Internet environment with resource dynamic nature, we extend the interface automata to timing and resource interface automata and formally model the Internetware system resource using the extended interface automata. We present an algorithm for verifying timing resource constraints and validate the trustworthiness of Internetware adaptation under the Internet environment with resource dynamic nature by checking whether the behavior of the component compositional system satisfies resource constraints within the time bound. Our future research work is to develop a corresponding prototype for automatic analytic and validating tools.

Acknowledgements

This research is supported by the NSFC of China (No. 71371012) and the Social Science and Humanity Foundation of the Ministry of Education of China (No. 13YJA630098). The authors would like to thank the anonymous reviews and the editor for their helpful comments and suggestions.

References

1. H. Mei and J. Lü, "On Environment-Driven Software Model for Internetware," Internetware, pp. 33-69, Springer, Singapore, 2016
2. H. Mei and J. Lü, "Internetware: A Shift of Software Paradigm," Internetware, pp. 3-17, Springer, Singapore, 2016
3. T. Xie, A. V. Hoorn, H. Wang, and I. Weber, "Introduction to the Special Issue on Emerging Software Technologies for Internet-based Systems: Internetware and DevOps," *ACM Transactions on Internet Technology* (*TOIT*), Vol. 18, No. 2, pp. 13, 2018
4. L. de Alfaro, T. A. Henzinger, and M. Stoelinga, "Timed Interfaces," pp. 208-122, Vinena: Springer-Verlag, 2002
5. A. Chakrabarti, L. de Alfaro, and T. A. Henzinger, "Resource Interfaces," pp. 117-133, Berlin: Springer-Verlag, 2003
6. J. Hu, Z. Q. Huang, D. Cao, and B. F. Xu, "Formal Analysis and Verification of Resource Adaptability for Internetware," *Journal of Software*, Vol. 19, No. 5, pp. 1186-1200, 2008
7. J. Zhang and H. Lei, "Algorithm for Computing Reliability Evolution of Internetware," *Journal of Southwest Jiaotong University*, Vol. 49, No. 2, pp. 311-318, 2014
8. X. Q. Xie, J. A. Zhou, D. A. Zhang, and Q. J. Xie, "Research on Trust Model of Internetware based on Complex Network," *Journal of Frontiers of Computer Science and Technology*, Vol. 10, No. 1, pp. 56-64, 2016
9. Z. Wei, M. Fan, M. Song, H. Wang, and Z. Zhang, "Analysis on the Technology of the Internetware Comprehensive Testing," Advances in Computer Science and Ubiquitous Computing, pp. 573-578, Springer, Singapore, 2017
10. E. D. L. Vitrand, X. Zuo, H. Yu, and S. Zheng, "An Approach to Evolving Internetware Application," in *Proceedings of International Symposium on Advances in Electrical*, *Electronics and Computer Engineering*, 2017
11. S. Y. Bao and Z. Q. Wang, "Business Consistency Verification Approach of Internetware Evolution," *Computer Engineering*, Vol. 37, No. 17, pp. 29-31, September 2011
12. Q. Xia and Z. Q. Wang, "Formal Analysis and Verification of Randomness Resources for Internetware," *Journal of Computer Applications*, Vol. 32, No. 11, pp. 3067-3070, November 2012
13. H. A. Sang, M. Q. Zhang, and J. Tang, "A Validation Method of Simulation Component Interface Design based on Reachability Chart," *Computer Simulation*, Vol. 27, No. 4, pp. 75-79, April 2010
14. M. Ben-ari, "Principles of the Spin Model Checking," Springer-Verlag, London, 2008
15. G. Honlzmann, "Spin Model Checker: Primer and Reference Manual," Addison-Wesley Publishing Company, Boston, USA, January 2004
16. L. Tan and H. W. Zeng, "Interface Automata-based Verification for Web Applications," *Computer Engineering and Applications*, Vol. 45, No. 3, pp. 70-73, March 2009

Appendix

Algorithm 1： Algorithm 1 checks whether a compositional system is satisfied by resource and timing constraints within a specified time bound.

Proof：

Let M be a reachable graph constructed by timing and resource interface automata, $X(node)$ is any node in the reachable graph M, where X_i is the number of resources used for state i, k_i is the number of resources provided by the environment for state i, t_i is time point, and (x_i, y_i) is the time limited by obtaining resources for state i.

(1) If $\exists X_i > k_i$ and $t_i \notin (x_i, y_i)$, then there exists a state that does not meet resource constraints and time limited. Therefore, M does not meet the environmental constraints.

(2) If $\exists X_i \le k_i$ and $t_i \notin (x_i, y_i)$, then there exists a state that satisfies resource constraints but does not meet the time limited. Therefore, M satisfies the resource constraint but does not meet the time limited.
(3) If $\exists X_i > k_i$ and $t_i \in (x_i, y_i)$, then there exists a state in M that does not meet the resource constraint but satisfies the time limit. Therefore, M does not satisfy the resource constraint but meets the time limited.
(4) If $\forall X_i \le k_i$ and $t_i \in (x_i, y_i)$, then any state in M can satisfy the resource constraint and time limited. Therefore, M satisfies the environment constraint.

Prof. Wang Zhongqun is from the School of Management Engineering, Anhui Polytechnic University, Wuhu, Anhui, China. His research interests include software engineering and machine learning.

Mr. Li Jun is currently pursuing his Ph.D. at the College of Command Information Systems, Army Engineering University of PLA, Nanjing, China. He is also as a lecturer at the School of Computer and Information, Anhui Polytechnic University, Wuhu, China. His research interests include software engineering, UAV ad hoc network, software-defined wireless networking, and Vehicular Ad Hoc Network.