Formal Specification and Verification of Timing Behavior in Safety-Critical IoT Systems

Yangli Jia, Zhenling Zhang, Xinyu Cao, and Haitao Wang

1 Introduction

The Internet of things (IoT) emerges as a common platform and service for consumer electronics [1]. IoT systems can be deployed into safety-critical missions such as defense, traffic, process control, environmental control, automotive systems, medical service, etc., and a failure in the temporal aspect in these systems can be as critical as one in the functional aspect and many, directly affecting the environment and lives of people [2]. To efficiently guarantee the high quality of these safety-critical IoT systems, it is necessary to clearly model, visualize, and verify the systems' interaction behavior before deploying them.

Formal modeling methods have the characteristics such as consistent, concise, unambiguous, and precise clarity, and we can also visualize and verify the behaviors based on the formal specification. Therefore it has great significance to improve accuracy, reliability, security of the systems by formal modeling, visualization, and verification of the behavior of complex IoT systems [3].

Many methods have been presented for modeling systems' interaction and other behavior properties [4]. These methods can be divided into two different categories. The first set of behavior specification methods can be called automata theory-based methods, such as duration automata [5], timed automata [6], timed I/O automata [7], and timed interface automata [8]. Timed automata has clock variable to express timing constraint information, while duration automata gives timing

Y. Jia · Z. Zhang (✉)
School of Computer Science & Technology, Liaocheng University, Liaocheng, China
e-mail: jiayangli@lcu.edu.cn; zhangzhenling@lcu.edu.cn

X. Cao · H. Wang
China National Institute of Standardization, Beijing, China
e-mail: caoxy@cnis.ac.cn; wanght@cnis.ac.cn

H. R. Arabnia et al. (eds.), *Advances in Software Engineering, Education, and e-Learning*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70873-3_32

constraints by binding a simple upper bound and lower bound for each transition. The other set of component behavior specification methods is usually based on process algebras, such as timed CSP [9], hybrid CSP [10], and behavior protocol [11]. All of these methods are easy to learn and usually supported by automated verification, but as without considering structural aspect of components, such behavior modeling techniques focus only on the behavioral aspect and are unable to describe the interconnection structure of hierarchical component architecture which also influences the behavior. This is one of the reasons why these models are often considered unavailable and cannot describe the time constraint information in IoT systems [12].

In this paper, we use formal method to model the interactive actions and give a visualization of these interaction behaviors. The formal model can be used easily to specify and verify IoT systems' interaction behavior and timing constraint information.

The rest of this paper is organized as follows. Section 2 focuses on the specification language of enhanced time behavior protocol. Composition and verification of timing behavior are discussed in Sect. 3. Section 4 gives an application example. The last section concludes the paper with the future work.

2 Specification Language for Modeling of IoT Systems' Behavior

2.1 Behavior Protocol

Behavior protocol [11] is a formalism used to describe abstract model of software systems by a set of admissible sequences of method calls. In behavior protocol, every method call or a return from a method call forms an event which can be denoted by event tokens. For a method named m, event tokens !mˆ, ?mˆ, !m#, and ?m# stand for emitting a method call, accepting a method call, emitting a return, and accepting a return. We can use the basic operators (eg. ";", "+", and "*"), the enhanced operators (eg. "|", "||", and "/"), and the composed operators (eg. "$\cap_x$", "|T|") to construct a behavior protocol in a way similar to a regular expression. A sequence of event tokens denoting events occurs in a component of the system form a trace. Thus, the trace <!mˆ,?m#> describes the activity of a caller (emitting a method call followed by accepting the return), while the trace <?mˆ; !m#> denotes what the call does (accepting a call and emitting the return).

Behavior protocol uses regular-like expressions to specify systems' interaction behavior and provides clear support for behavior specification and formal reasoning about the correctness of behavior. In addition, it is easy to read and apply.

2.2 Enhanced Time Behavior Protocol

We extend behavior protocol to the area of IoT systems and the result is enhanced time behavior protocol (ETBP). ETBP programs model systems' interaction behavior as a sequence of timed communication events and timed internal events. Events in ETBP are bound with a timing constraint like duration automata and a 2-tuple time consumption constraint according to requirements of IoT applications. ETBP programs are easy to read and apply just as BP programs. Based on the advantages of both behavior protocol and duration automata, ETBP can be used easily to specify and verify IoT systems' interaction behavior and timing constraint information. The syntax and semantics of enhanced time behavior protocol owe much to timed CSP [9].

We can specify the time delay conditions and the time consumption constraint by means of timed event tokens. Inspired by duration automata [5], we extend event tokens by binding timing constraint and time consumption information on each event token needed. So timed event tokens take the forms of !m[t_1][t_2,t_3]^, ?m[t_1][t_2,t_3]^, !m[t_1][t_2,t_3]#, and ?m [t_1][t_2,t_3]#.

For example, !interface.method[t_1][t_2,t_3]^ stands for that invoking component of IoT system emit a method call request within t_1 time units delay and the event's time consumption interval is [t_2,t_3], while ?interface.method[t_1][t_2,t_3]^ means the request must be accepted by the target component of IoT system within t_1 time units. !interface.method[t_1][t_2,t_3]# stands for the target component emits a response delayed for at most t3 time units, and ?interface.method[t_1][t_2,t_3]# means the invoking component accepts the response within t_4 time units. All the invoke and acception events must be done in time interval [t_2,t_3].

The time consumption constraint is represented as a 2-tuple [a,b], where a,b $\in$ N and a $\leq$ b. The relative time consumption interval [a,b] is a time consumption interval measured with respect to some reference time consumption instant. We represent it by a natural number. The number starts from 0 and increases by 1 each time when a new tick is generated.

Time consumption constraints will be used as guards for enhanced time behavior protocol.

Basic operators of ETBP are defined in classical regular expressions.

Enhanced operators in ETBP include the parallel operators "|" and "||", the restriction operator "/", the timeout operator, the delay operator "Idle," the guard operator "when," and the reset operator and the "Stop" operator.

If we use A, B denotes a protocol. Timed BP can be defined by the following Backus-Naur form:

$$\begin{aligned} P ::= {} & A; B \mid A + B \mid A* \mid A \triangleright_t B \mid A \mid B \mid A \mid \mid B \mid A/G \mid \text{Reset (t)} \mid \text{Idle (t)} \mid \\ & \text{When} \mid \text{Stop} \mid A \cap_x B. \end{aligned}$$

Basic operators in ETBP include the sequencing operator ";", the alternative operator "+", and the repetition operation "*".

- A;B represents a succession of protocol A by protocol B. The set of event traces formed by A;B are a concatenation of a trace generated by A and a trace generated by B.
- A+B represents an alternative of protocol A and protocol B. The set of event traces formed by A+B are generated either by A or by B.
- A* represents an repetition of protocol A., and it is equivalent to NULL + A + (A;A) + (A;A;A) + ... where A is repeated any finite number of times.

Then we give the enhanced operators including the parallel operators "|" and "||" and the restriction operator "/".

- A|B represents the "and-parallel" of protocol A and protocol B. The set of event traces formed by A|B are an arbitrary interleaving of event tokens of traces generated by A and B.
- A||B represents the "or-parallel" of protocol A and protocol B. It stands for A + B + (A|B).
- A/G represents the "restriction" of protocol A. The event tokens not in a set G are omitted from the traces generated by A.

Besides operators in behavior protocol, we give other time-related operators to ETBP to construct more complicated protocols. These operators include the timeout operator, the delay operator "Idle," the guard operator "when," the reset operator, and the "Stop" operator.

- $A \triangleright_t B$ expresses the timeout operation of protocol A and protocol B. It may execute any events that A may perform before time t, but if a timeout occurs, it will execute events that B may perform.
- When operator takes the form (when b A) which means that it will perform events trace generated by A if b is true, otherwise the events trace generated by A cannot be chosen for execution.
- Idle(t) is the delay operator. It does nothing but waits t time units.
- Reset(t) operator just reset the clock variable *t* to 0.

Stop can be used to specify a component behavior which does nothing except terminate.

2.3 Example of Enhanced Time Behavior Protocol

Consider the protocol ?a;(!p[2][8, 9]+!q[2][8, 9])$\triangleright_2$!b[2] [8, 9]||?c. It contains event tokens ?a,!p[2][8, 9],!q[2][8, 9],!b[2] [8, 9],?c and the operators ;, +, $\triangleright_2$, and ||. (In the examples here, we omit event suffixes "^" and "#" and connection names for simplicity.) The protocol generates traces including, for instance, <?a;!p[2][8, 9] $\triangleright_2$!b[2][8, 9]>, <?a;!q[2][8, 9] $\triangleright_2$!b[2][8, 9]>, <?a;?c;!p[2][8, 9]$\triangleright_2$!b[2][8, 9]>, and <?a;!p[2] [8, 9] $\triangleright_2$!b[2][8, 9];?c>. The trace <?a;!p[2][8, 9] $\triangleright_2$!b[2][8, 9]> starts with ?a, followed by !p[2][8, 9] $\triangleright_2$!b[2][8, 9], which means emitting the

method call p within 2 time units delay, and if timeout occurs, it will emit the method call b within 2 time units. All the invoke and acception events must be done in time interval [8, 9]

3 Composition and Visualization of ETBP

To better support for complex real-time component-based systems' development, we must provide compatibility verification theory and automation tools for the ETBP model.

3.1 Composition of Enhanced Time Behavior Protocol

For behavior protocol P and behavior protocol Q, we use P $\cap_x$ Q to give the definition of composition of enhanced time behavior protocol P and Q. Their composition must meet the timing constraint information of the application. In the case of composition, the time behavior protocol P or Q sends an event, the time behavior protocol Q or P receives an event, the time behavior protocols P and Q interact with the event, and internal events are generated.

If their composition meets the timing constraint information, any appearance of !interface.method[t][t_2,t_3], ?interface.method[t][t_2,t_3], resp. ?interface.method[t][t_2,t_3], and !interface.method[t][t_2,t_3], as a result of the interleaving, is merged into τinterface. method[t] in the resulting trace, and the invoke and acception events must be done in time interval [t_2,t_3] if every event has a time consumption constraint.

In an ideal case, the event perfectly matches, that is, the time behavior protocol P or Q executes the "send an event," and the corresponding behavior protocol Q or P exactly executes the "receive an event," the time consumption constraint also satisfied, and then the two time behavior protocols P and Q interact with the event to generate internal events.

For example:the enhanced time behavior protocols P = !tm.begin[t_1][t_3,t_4]; (!tm.commit[t_1][t_3,t_4]+!tm.rollback[t_1][t_3,t_4]) and Q = ?tm.begin[t_1][t_3,t_4]; ?tm.rollback[t_1][t_3,t_4], P$\cap_x$Q create the path τtm. begin; τtm. rollback. The combined enhanced time behavior protocol does not contain events tm. commit, because the right side of the combination operator requires that events rollback must occur and events rollback and commit can only be executed by one of them.

In order to describe composite operations accurately, we give the formal operational semantics of composite operations, and the interaction between P and Q needs to be considered. Suppose A and B are event sets of time behavior protocol P and Q, respectively. In the case of interaction, protocol P or Q executes the sending operation, and the protocol Q or P executes the receiving event. P and Q interact

with the event on this way, and internal events are generated in the case of combined operation.

From an evolutionary perspective,

$$\frac{P \underset{t}{\rightarrow} P', Q \underset{t}{\longrightarrow} Q'}{P \cap_X Q \underset{t}{\longrightarrow} P' \cap_X Q'}$$

From a migration perspective,

$$\frac{P \overset{(t,!a)}{\longrightarrow} P', Q \overset{(t,?a)}{\longrightarrow} Q'}{P \cap_X Q \overset{(t,!a \wedge ?a)}{\longrightarrow} P' \cap_X Q'} [!a \in A, ?a \in B, a \in X]$$

3.2 Composition and Verification of Behavior Protocols

We give the composition and verification algorithm of timing behaviors as follows. Algorithm input: enhanced time behavior protocol P, Q, event set X, Algorithm output: the composition result of P,Q. (1) read the two protocol P, Q, classified the events in P,Q into the sets $S_{P,\,prov}$, $S_{P,\,req}$, $S_{Q,\,prov}$, $S_{Q,\,req}$(2) while not end of protocol P or Q, (3) if there is a call event ?mˆ[t][t_2,t_3] or !m#[t][t_2,t_3] in P or Q, (4) then traverse Q or P to find if there is any event m in the form of ?mˆ[t][t_2,t_3] or ?m#[t][t_2,t_3], generates the composition traces tc = !mˆ[t][t_2,t_3]?mˆ[t][t_2,t_3] , or tc = !m# [t][t_2,t_3]?m#[t][t_2,t_3], $m \in X$(5) T(c) =Utc_i(6) traverse all T (c), (7) any $tc \in T(c)$ (8) while tc is not terminated (9) if there is trace tc1 = !mˆ[t_1][t_3,t_4]?mˆ[t_2][t_3,t_4], or, tc2 = !m#[t_1][t_3,t_4]?m#[t_2][t_3,t_4],$m \in X$(10) then (11) if there is overlap of [t_1], [t_2] and done in [t_3,t_4] (12) then combined as $\tau m[t]$ (13) else output the path, and invalid timed activity error (14) if there is $m \in (S_{P,\,prov} \cup S_{Q,\,prov}) \wedge m \in X$(15) then output the path, and invalid timed activity error (16) if there is $m \in (S_{P,\,req} \cup S_{Q,\,req}) \wedge m \in X$(17) then output the path, and stop forward error (18) tc $\leftarrow$ tc$'$, tc$'$ $\in T(c)$, goto (8)

Obviously, the algorithm gives the composition result, and compatibility of two protocol is verified by determining the related compatible errors in composition. Based on the composition algorithm, we can visualize the process and the result.

Based on the LTSA tool [13] which has an extensible architecture allowing extra features to be added by means of plugins, we have developed an integrated tool named ETBPSV for specifying and verifying IoT systems' behavior.

4 Application of ETBP

To demonstrate the specification model described above, consider the case of a boiler pressure control real-time system. The boiler pressure control real-time system consists of four components including pressure sensor, pressure monitor, pressure controller, and alert. The pressure monitor component acquires pressure data from the pressure sensor component every 18 ms and sends information to the pressure controller component within 2 ms to request the issue of cooling or warming if the pressure is too high or too low. The controller responds to the information within 1 ms. If error occurs, the system will give an alarm and stop running within 1 ms.

We give the formal specification of enhanced time behavior protocol for the components in the system as follows:

```
Pressure_sensor:
?pressure^;!pressure#;
Pressure_monitor:
(!pressure^;?pressure#;
(
!low^[2]▷2 (!alert^;?alert#;Stop)[1];?low#; Idle(18-t)
+!high^[2]▷2 (!alert^;?alert#;Stop)[1];?high#;Idle(18-t)
+Idle(18);
)
)*;
where t is the delay time for the pressure monitors sending
    request information of cooling or warming to the
    controller, and t<2.
Pressure_controller:
?low^;!low#[1]+?high^;!high#[1];
Alert:
(?alert^;!alert#) or ?alert.
The four sub-components can be combined into a
    composition-component (system/subsystem), which behavior as:
(τpressure^;τpressure#;
(
τlow^[2]▷2 ( τalert^;τalert#;Stop)[1];τlow#; Idle(18-t)
+τhigh^[2]▷2 ( τalert^;τalert#;Stop)[1];τhigh#;Idle(18-t)
+Idle(18);
)
)*;
```

By using the time-related operators in ETBP, components' real-time behavior can be formally specified easily and precisely in designing component-based IoT systems. Obviously, combining the advantages of both simplicity and practicality, ETBP has more powerful description ability and can be used easily to specify real-time components' behavior and timing constraint information. The behavior protocols of real-time sub-components can be composed together to build high-level protocols based on the composition definition. And based on the formal specification, we can analyze and verify the timeliness, safety, and other trustworthiness properties of component-based IoT systems.

We input the enhanced time behavior protocol of the three components into ETBPSV. After compiling, we can see the states migration diagram of each protocol as shown in Figs. 1, 2, and 3, respectively.

The behavior protocol composition migration diagram of pressure sensor and pressure monitor is shown in Fig. 4.

As the composite protocol will send out event !tem_low^[2], when it combines with time behavior protocol of pressure controller, the corresponding response event

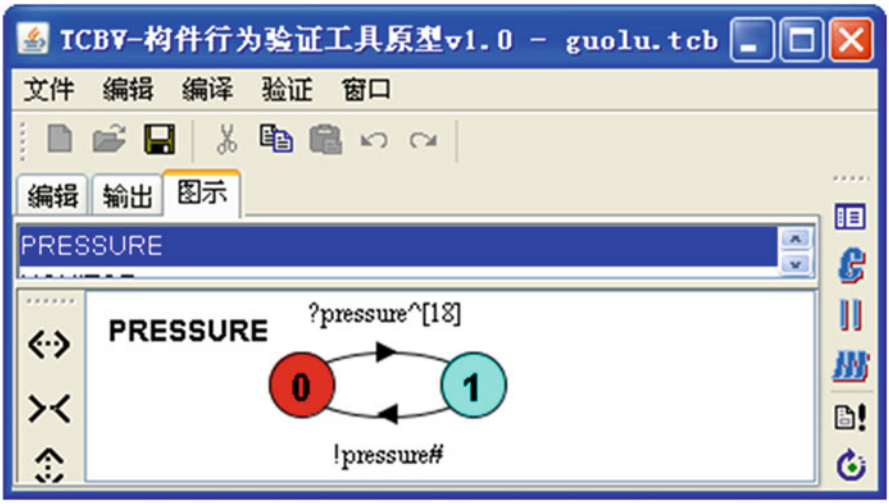

Fig. 1 States migration diagram of pressure sensor's protocol

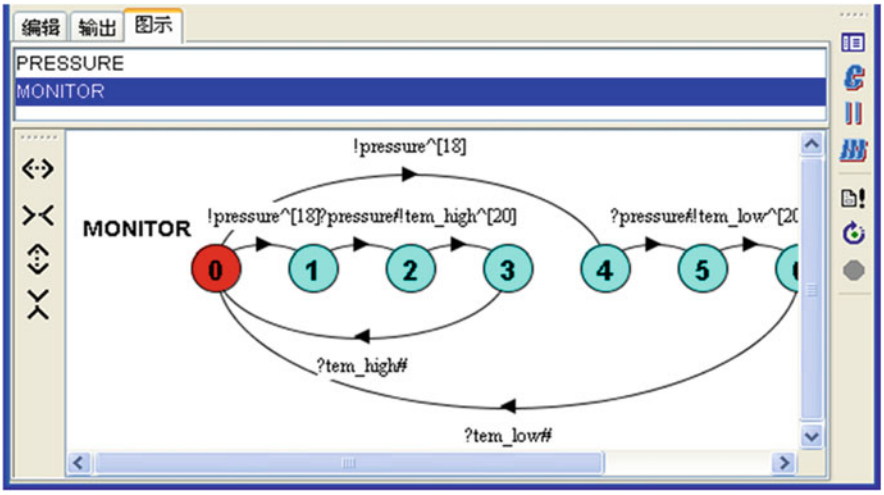

Fig. 2 States migration diagram of pressure monitor's protocol

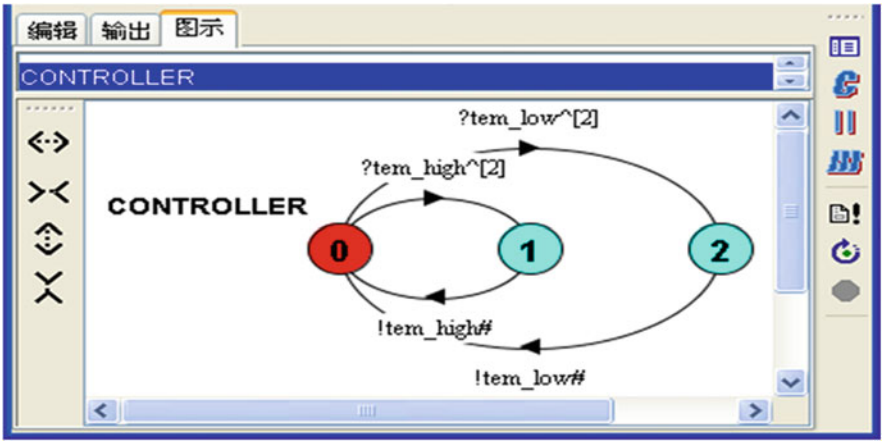

Fig. 3 States migration diagram of pressure controller's protocol

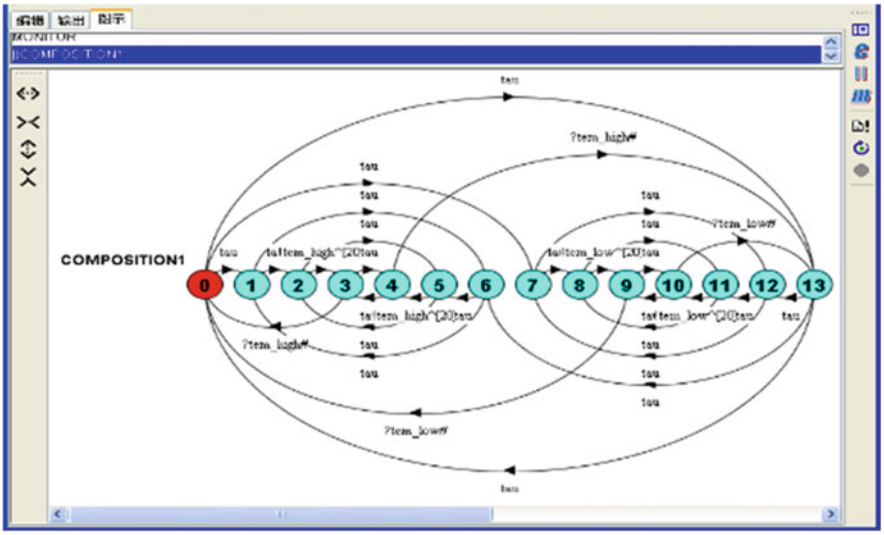

Fig. 4 States migration diagram of the combined two protocols

?tem_high^[2] cannot be found; the combined result will give bad activity prompt, as shown in Fig. 5.

Three behavior protocols can be combined into one, and the graphical results can be displayed using ETBPSV as Fig. 6. It can be seen that the diagram is very complex. Therefore, it is unrealistic to analyze the results of time behavior protocol combination manually. We can formally verify in ETBPSV as shown in Fig. 7.

We have experimentally specified several specifications of IoT systems modeled by enhanced time behavior protocol, and these specifications were visualized

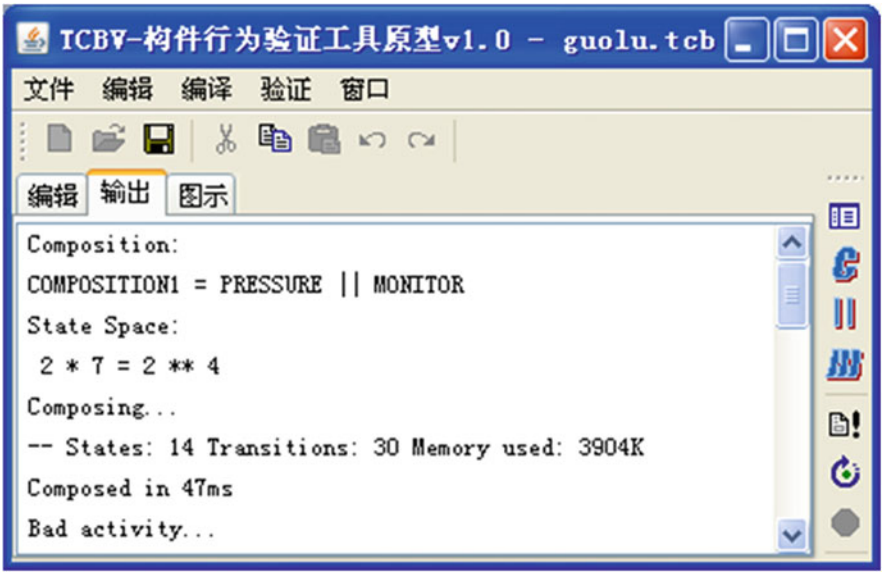

Fig. 5 Verification of the combined two protocols

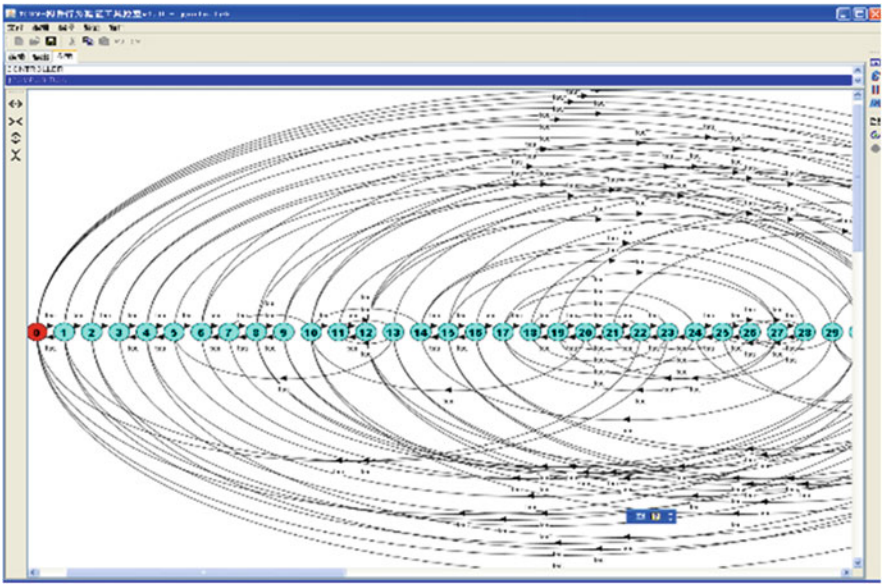

Fig. 6 Migration diagram of the combined three protocols

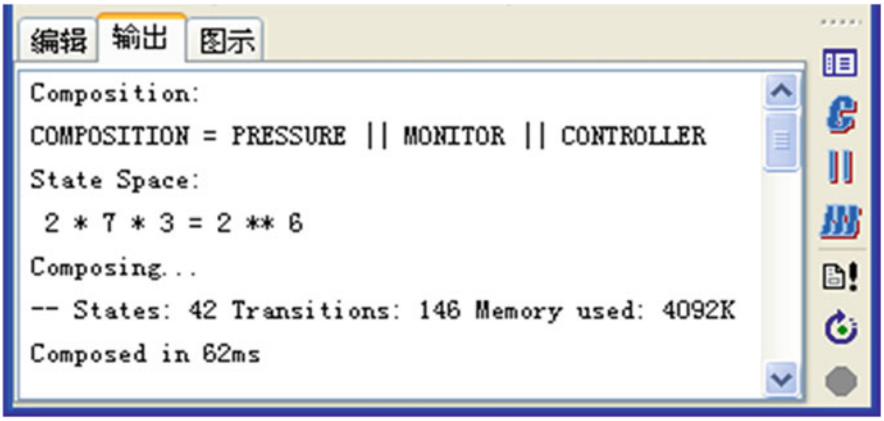

Fig. 7 Verification result of the composition

and verified using ETBPSV, and some errors about timeliness, safety, and other trustworthiness properties were found effectively. But with the complex increasing of ETBP model, state space explosion problem is becoming significantly aggravated and affect the efficiency of visualization and verification seriously.

5 Conclusions and Future Work

We presented a formal specification method of timing interaction behavior in IoT systems. In addition, we developed a more efficient automatic verification framework based on enhanced time behavior protocol and gave an example to show how the method can be used. Combining the advantages of both simplicity and practicality, the enhanced time behavior protocol has more powerful description ability and can be used easily to specify real-time interaction behavior and timing constraint information which provide a rich base for further application of formal methods.

As future work related to timing behavior specification and verification, we intend to focus on (1) giving the semantics of operators in enhanced time behavior protocol and (2) dedicating to the research of state space reduction algorithm.

Acknowledgments This work is supported by the National Natural Science Foundation of China under Grant No. 81973695 and Soft Scientific Research Project of Shandong Province under Grant No. 2018RKB01080.

References

1. A. Čolaković, M. Hadžialić, Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. Comput. Netw. **144**, 17–39 (2018)
2. K. Hofer-Schmitz, B. Stojanović, Towards formal verification of IoT protocols: A review. Comput. Netw., 107233. ISSN 1389-1286. (2020). https://doi.org/10.1016/j.comnet.2020.107233
3. S. Tang, D.R. Shelden, C.M. Eastman, et al., A review of building information modeling (BIM) and the internet of things (IoT) devices integration: Present status and future trends. Autom. Constr. **101**(May), 127–139 (2019)
4. P. Fremantle, B. Aziz, Deriving event data sharing in IoT systems using formal modelling and analysis, Internet of Things. **8**, 100092, ISSN 2542-6605 (2019). https://doi.org/10.1016/j.iot.2019.100092
5. D.V. Hung, B.V. Anh, Model checking real-time component based systems with blackbox testing, in Proceedings of IEEE RTCSA'05, Washington, DC, 2005, pp. 76–79
6. F. Heidarian, J. Schmaltz, F.W. Vaandrager, Analysis of a clock synchronization protocol for wireless sensor networks. Theor. Comput. Sci. **413**(1), 87–105 (2012)
7. D.K. Kaynar, N. Lynch, R. Segala, F. Vaandrager, *The Theory of Timed 1/O Automata[R]* (MIT Laboratory for Computer Science, Cambridge MA, 2004)
8. L. de Alfaro, T.A. Henzinger, M. Stoelinga, Timed interfaces[C], in *Proceedings of the Second International Workshop on Embedded Software*, (Springer, Berlin, 2002), pp. 108–122
9. J. Davies, S. Schneider, A brief history of timed csp. Theor. Comput. Sci. **138**(2), 243–271 (1995)
10. H. Jifeng, From CSP to hybrid systems, in *"A Classical Mind," Essays in Honour of C.A.R. Hoare*, International Series in Computer Science, ed. by A. W. Roscoe, (Prentice Hall, 1994), pp. 171–189
11. F. Plasil, S. Visnovsky, Behavior protocols for software components. IEEE Trans. Softw. Eng. **28**(11), 1056–1076 (2002)
12. L. Brim, I. Cerna, P. Varekova, B. Zimmerova, Component-interaction automata as a verification-oriented component-based system specification. SIGSOFT Software Engineering Notes 31 **31**(2) (2006)
13. J. Magee, J. Kramer, *Concurrency-State Models and Java Programs* (Wiley, 1999)