Formal verification of dynamic hybrid systems: a NuSMV-based model checking approach

Zhi Xu[1,a,], Deming Zhong[1,b], Weigang Li[2,c], Hao Huang[2,d] And Yigang Sun[3,e]*

[1]Beihang University, XueYuan Road No.37,HaiDian District, BeiJing, China
[2]Hangfa Control System Research Institute of China, LiangXi Road No.37, BinHu District, WuXi, China
[3]Civil Aviation University Of China, North Way of Road, Dong Li District,Tian Jin,China
[a]332999245@qq.com; [b]zhongdeming@buaa.edu.cn; [c]bailingke@vip.sina.com; [d]1316247928@qq.com; [e]caucsyg@163.com

Abstract. Software security is an important and challenging research topic in developing dynamic hybrid embedded software systems. Ensuring the correct behavior of these systems is particularly difficult due to the interactions between the continuous subsystem and the discrete subsystem. Currently available security analysis methods for system risks have been limited, as they rely on manual inspections of the individual subsystems under simplifying assumptions. To improve this situation, a new approach is proposed that is based on the symbolic model checking tool NuSMV. A dual PID system is used as an example system, for which the logical part and the computational part of the system are modeled in a unified manner. Constraints are constructed on the controlled object, and a counter-example path is ultimately generated, indicating that the hybrid system can be analyzed by the model checking tool.

1 Introduction

Hybrid systems [1], also known as hybrid dynamic systems, are designed using interactions between the continuous subsystem and the discrete subsystem. The dynamic characteristics of continuous subsystems evolve with time, while the dynamic evolution of discrete subsystems is driven by events. Hybrid systems are found in a wide range of embedded system applications, including complex industrial control systems. In such systems, the dynamic evolution of the physical layer is described by the continuous computation of successive variables and the high-level coordination optimization process is characterized by symbolic operations and discrete monitoring decisions.

There are numerous challenges involved with the security analysis of dynamic hybrid systems [2]. First, the security attributes of such systems need analysis that combines comprehensive knowledge of control disciplines, computer science, mathematics, as well as security. Second, the interaction of the continuous and discrete subsystems adds complexity to the analysis problem. When a system is relatively simple, such as risk factors including only a single category, or a single system component without heterogeneous composition,

* Zhi Xu: 332999245@qq.com

or a linear time-invariant system, it is relatively straightforward to find and analyze the system risks. However, when analyzing the security of a complex system, due to heterogeneous component coupling, multi-class risk factors coexistence, nonlinear dynamic operation, and system state explosion, the existing security analysis techniques have been inadequate. The reason is that, when the two components of the hybrid system are analyzed separately, while ignoring the interaction of the two components, or the analysis of the interaction is only manually analyzed by experience, then the important system risks must be left out.

This work proposes a new approach to analyzing the security of dynamic hybrid systems that is based on a formal method. Formal methods are an effective means of logical consistency verification of the system, which can ensure the adequacy of verification. Formal methods include theorem proving and symbolic model checking. Model checking is a state traversal-based verification technique, iterating through all the system states to determine if the system meets the desired characteristics. In general, the inputs to a model checking tool include two parts: the finite state machine model, and the system features constrained by temporal logic expressions.

In this paper, the logical part and the computational part of a dual PID system are modeled in NuSMV [3]. Constraints are constructed on the controlled object, and a counter-example path is ultimately generated, indicating that the hybrid system can be analyzed by the model checking tool.

2 Case study

2.1 Preliminary

2.1.1 NuSMV

The research on model checking is mainly embodied in the development of model checking tools. At present, there are many open source model detection tools, among which NuSMV and SPIN [4] are the most widely used ones. NuSMV was developed by Dr. McMillan from Carnegie Mellon University, primarily through the reconstruction and performance optimization of SMV (symbolic model verifier) [5]. The NuSMV kernel is based on binary decision diagrams (BDDs) and is the first model detection tool based on a binary decision graph. NuSMV is completely open sourced and has good scalability; consequently numerous customized tools based on NuSMV have been presented. NuSMV is often used for reliability and security verification of industrial designs, and the NuSMV core is often used as a core component of other tools because of its powerful core capabilities and good scalability. NuSMV has many features, including interactive, call operations, variable analysis, partition, system characteristics test, as well as SAT (Boolean Satisfiability Problem). NuSMV is not only compatible with the SMV batch task, but also provides a set of textual interactive interfaces. Users can type a variety of instructions to perform relevant operations and calculations. NuSMV can support computational tree logic (CTL) and linear temporal logic (LTL). . When validating system features, the users can specify the path they are interested in and check the value of the associated variable on that path. When NuSMV discovers that the system has a path conflicting with the characteristics, the counter-example path will be fed back to the user.

2.1.2 Overview of PID control system

PID control is one of the earliest developed control strategies, and is widely used in industrial control due to its simplicity, robustness, as well as high reliability. A typical single PID closed-loop control system is shown in Figure 1.

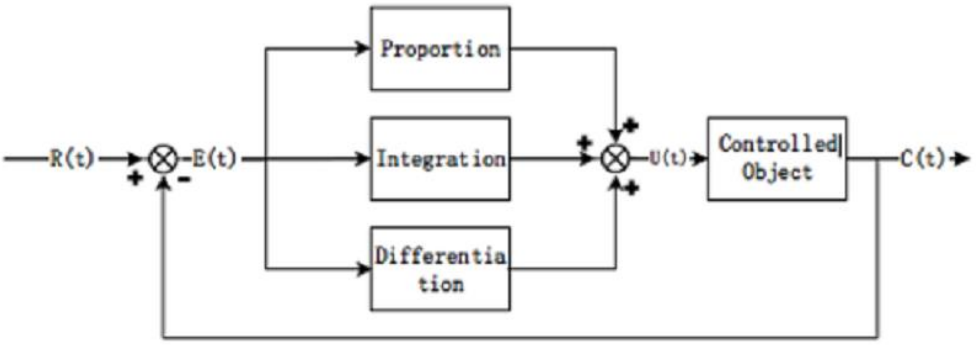

Fig. 1. Typical single PID closed-loop control system

According to the control deviation of the given value r(t) and the actual output value c(t), the PID controller computes the control value using the linear combination of Proportion (P), Integration (I) and Differentiation (D) to control the object. Its control rule obeys Equation 1:

$$u(t) = K_P\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D\frac{de(t)}{dt}\right] \quad (1)$$

where K_P the proportional coefficient, T_I is the integral time constant and T_D is derivative time constant.

2.2 Model construction

2.2.1 Models and parameters setting

To illustrate that hybrid systems can be analyzed using model checking tools, we select a double PID control system as an example to model and analyze. The specific model is shown in Figure 2. In this system, the two PID controller switching is the logical part, while the proportional, integral and differential calculation parts in each PID are the continuous part.

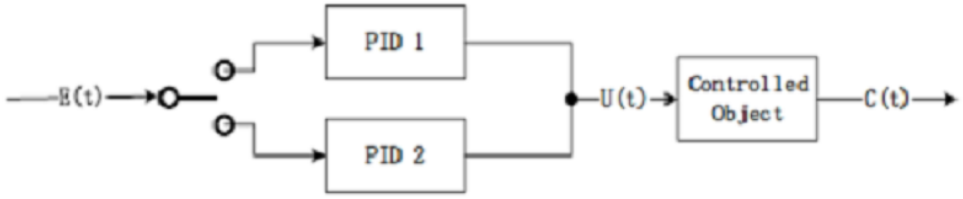

Fig. 2. Typical single PID closed-loop control system

Because the focus of this paper is on the logic and calculation part of the hybrid system, for a more direct and effective problem description, as well as to avoid unnecessary interference, ideal processing parameters are assumed. The time t is set as follows: when the time is in 0-3 seconds, the first PID controller is used; when the time is in 4-6 seconds, the second PID controller is used (Equation 2).

$$u(t) = \begin{cases} K_{P1}\left[e(t) + \frac{1}{T_{I1}}\int_0^t e(t)dt + T_{D1}\frac{de(t)}{dt}\right] & (0 \le t \le 3) \\ K_{P2}\left[e(t) + \frac{1}{T_{I2}}\int_0^t e(t)dt + T_{D2}\frac{de(t)}{dt}\right] & (4 \le t \le 6) \end{cases} \quad (2)$$

where $e(t) = 3t^2$, $K_{P1} = 1/3$, $T_{I1} = 1/3$, $T_{D1} = 1/2$, $K_{P2} = 1/3$, $T_{I2} = 1/3$, and $T_{D2} = 1$.

2.2.2 Model transformation

Based on the above description of the simplified models, the NuSMV representation is defined. The following transformation result as well as the transformation process description: some key script are shown in Table 1. (comments are the transformation process description).

Table 1. Key script about the model transformation

```
MODULE main
VAR
t : 0..6;  -- running time of the whole system is 6s, and the step is 1s
P1 : 0..9;  -- the value range of the proportional part of the first PID controller
I1 : 0..27;  -- the value range of the integral part of the first PID controller
D1: 0..6;  -- the value range of the differential part of the first PID controller
P2 : 0..36;  -- the value range of the proportional part of the second PID controller
I2 : 0..216;  -- the value range of the integral part of the second PID controller
D2: 0..6;  -- the value range of the differential part of the second PID controller
u : 0..258;  -- the output value range of the controller
y : Boolean;  -- controlled object is set to Boolean
……
next(t) := case   -- the state transition paths of time
  (t >= 0) & (t <= 5) : t + 1 ;
  t = 6 : 0;
TRUE : t;
……
next(u) := case  -- the state transition paths of output u
(t >= 0) & (t < 3) : next(p1) + next(i1) + next(d1);
(t >= 3) & (t <= 6) : next(p2) + next(i2) + next(d2) mod 259 ;
 u = 258 : 258;
TRUE : u;
esac;
next(y) := case  -- the state transition paths of controlled object y
next(u) = 258 : TRUE;
next(u) < 258 : FALSE;
esac;
```

The constraints are constructed as follows.

CTLSPEC ! EF (y = TRUE)

Where y is the control object, which is a Boolean logic value.

The NuSMV model can be interpreted as a switch: when the state is TRUE, the switch is open; otherwise the switch is closed. Therefore, the above constraints on the model can be explained as follows: there is no state transition path, such that the switch is open.

2.2.3 Implementation verification

The output obtained after the NuSMV terminal loading the model script is shown in Figure 3.

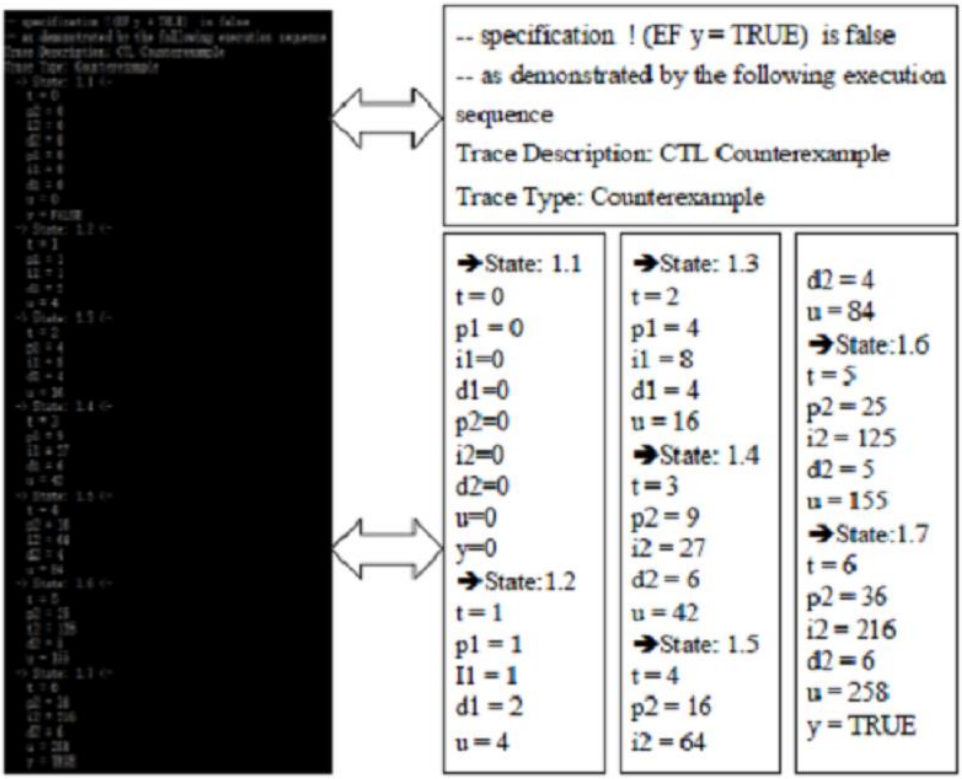

Fig. 3. The output after the NuSMV terminal loading the model script

Because NuSMV can support CTL/LTL logic, when the system characteristics are verified, the path that the users are interested in can be specified and the value of the associated variable on that path can be checked. When NuSMV discovers that the system has a path that conflicts with the characteristics, these counter-example paths are fed back to the users.

The results demonstrate that the system eventually leads to y=TRUE, i.e., the switch is opened, and a counter-example path is given to explain how each part of the state changes will lead to the final results of switch being opened.

3 Conclusion

In this paper, a model checking method is proposed to unify the logical part and the computational part of a dynamic hybrid system model. This provides a formal approach to the analysis of such systems, enabling their adoption into security critical embedded systems. A dual PID system is used as an example. The final results show that the two parts of the original separated process can be modeled together and the desired results can be obtained.

In the future, more complex hybrid systems need to be further explored, so that the proposed method can be more applicable, and the model expression can be more complete.

This research was supported by grants from Civil Aviation Joint Funds Established by National Nature Science Foundation of China and Civil Aviation Administration of China(No.U1533201), from a project of Ministry of Industry and Information Technology of China (No.JSZL2015601C008) and from the Major State Basic Research Development Program of China (973 Program) (No. 2014CB744904).

References

[1] KONG, S., GAO, S., CHEN, W., and CLARKE, E., 2015. dReach: δ-Reachability Analysis for Hybrid Systems. In International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems, 200-205.

[2] YAN, Q., YU, F.R., GONG, Q., and LI, J., 2016. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. IEEE Communications Surveys & Tutorials 18, 1, 602-622.

[3] CIMATTI, A., CLARKE, E., GIUNCHIGLIA, F., and ROVERI, M., 2000. NUSMV: a new symbolic model checker. International Journal on Software Tools for Technology Transfer 2, 4, 410-425.

[4] HOLZMANN, G.J., 1997. The model checker SPIN. IEEE Transactions on Software Engineering 23, 5, 279-295.

[5] MCMILLAN, K.L., 1992. The SMV system. Symbolic Model Checking 38, 4, 161–165.