A Hierarchy Framework on Compositional Verification for PLC Software

Litian Xiao[1,2] and Mengyuan Li
[1]*Beijing Special Engineering Design and Research Institute*
Beijing 100028, China

{xiao_litian & li_mengyuan2000}@sina.com

Ming Gu and Jiaguang Sun
[2] *School of Software, TNList, KLISS*
Tsinghua University
Beijing 100084, China
{guming & jgsun}@ tsinghua.edu.cn

***Abstract*—The correctness verification of embedded control software has become an important research topic in embedded system field. The paper analyses the present situation on correctness verification of control software as well as the limitations of existing technologies. In order to the high reliability and high security requirements of control software, the paper proposes a hierarchical framework and architecture of control software (PLC program) verification. The framework combines the technologies of testing, model checking and theorem proving. The paper introduces the construction, flow and key elements of the architecture.**

Keywords-PLCsoftware; compositional verification; hierarchy framework; verification architecture

I. INTRODUCTION

Programmable Logic Controller (PLC) is a kind of embedded system in automatic control system. PLC software is a core for controlling, monitoring or managing other devices. The program logic of PLC software controls the action circuits of control system. Now PLC software has larger scale and more complex functions so that its correctness is difficultly ensured. Its errors or bugs will lead to unpredictable or uncontrolled behavior of control system [1].

Limited fewer embedded testing means, PLC program cannot be directly tested and verified [2]. Now its testing generally needs to build the operating environment through the actual hardware and software, and sometimes even is required to complete in real environment. The testing is costly and cannot guarantee test coverage. Some boundary or wrong tests may cause controlled device fault, serious damages or even accidents. How to ensure the correctness of PLC software in safety-critical automatic control systems has become an important research in the field of embedded system.

The paper presents a hierarchy framework for compositional verification and strategies of correctness verification for PLC software. It mainly introduces the architecture and key factors of the hierarchy framework. The main work and achievements of the research are introduced.

II. RELATED WORK ON SOFTWARE VERIFICATION

To ensure software correctness, testing, formal or mathematic and systematic methods are used to avoid program bugs. The main methods are aimed at checking programs on the three levels of code, model and statute, which include:

- Testing methods: It finds and reproduces bugs in program while specific functional segments or code segments (test cases) are really executed in some particular scene or by input data.
- Verification methods: By means of a certain abstract mechanism, the program codes are transformed into specific mathematical representation. Then the correctness conclusion of a program is obtained by searching, reasoning, proof and other means based on mathematical representation.

Their technologies are testing, model checking and theorem proving and currently can be used for PLC program.

A. Testing Technology on Code Level

Software testing is an effective means to find program bugs. The found bugs are real existence in a program. The biggest drawback of testing is incomplete, i.e. it is difficult to traverse all possible execution paths for slightly complex program. It is difficult to design a systematic approach and use limited test cases for completely covering up to all codes.

For embedded software testing, hybrid prototype is mainly utilized by international research institutions such as NASA, Boeing, Israel AirForce. It creates a test environment by means of emulators, simulators, software modeling and injected host computer data. The hardware prototype components of hybrid prototype are directly connected to the target system and are given controlled response signals to that. The response signals are the same as the actual signals of real system and environment. Host computer injects testing data into target system based on software model, control model and processes. Such research is also a hot issue, but test cost is very high.

B. Model Checking Technology on Model Level

Verification method and test technology have strong complementarity. Also verification method is a hot research issue [3]. Model checking is a type of verification method based on model, searches state space and verifies specified nature. Generally function nature is safety property, liveness property and fairness property etc. [4] The method is reliable and complete for specified nature. Its biggest problem is "state space explosion". Although some scholars have studied model checking algorithms for infinite-state, currently these algorithms cannot be applied to general purposes.

This research is sponsored by NSFC Program (No.90718039, No.91018015 and No.60811130468) of China

978-1-4799-3279-5 /14/$31.00 ©2014 IEEE

Even if model checking obtains an affirmative conclusion, it is not able to ensure other properties such as non-functional properties (e.g. divided by zero, data overflow, etc.). Furthermore, model itself is not a program and is gotten by abstracted a real program codes. In the abstracted process, a constructed model is allowed to increase some non-existent actions in actual program, and not to delete any actions. Such abstracted mode possibly leads to false reports in the verification process--incorrectly report errors in actual program. The model must be further refined to eliminate false reports.

Now classic model checking tools include SPIN[5], NuSMV[6] and UPPAAL[7], etc.

C. Theorem Proving Technology on Statute Level

Another type of verification method is based on mathematical proof. These methods firstly describe the behavior characteristics of a system with a series of logical formulas. Then some properties are proven by means of a logical system (or proof tools), inference rules provided by deductive methods or falsification established goals.

Theorem proving method is very suitable for the verification of infinite state systems. Because most of the proving tools (e.g. PVS[8], COQ[9], Nqthm[10]) provide for higher-order logic which has the ability to describe infinite data structures, they are not sensitive to the size of state space.

However, there are some flaws on theorem proving methods. These methods don't have high automation degree and require a lot of manual operation. A high degree of expertise is required for theorem proving, and they need to be familiar with particular domain.

D. Verification Technology on PLC Program

The above-mentioned technologies can be used to verify PLC programs and have similar advantages and disadvantages. Now there are few verification technologies for PLC program on practical achievement.

Many researchers have studied how to do model-checking for PLC program. Some research directly converse specific PLC program into the input of model-checking tool. They demand single restriction model of PLC program, i.e. Boolean variables exist only in the program, and jump statements or multiple blocks or functions cannot [11]. Others abstract PLC program into models for model-checking tool. However, these models are somewhat abstract distortion and their checking scale is small [12]. The works can partly solve the problems of model-checking for some particular PLC programs, but there is a lack of reduced strategy of the state space size.

Combinatorial model-checking are widely studied and applied for decreasing state space size. The works depend on the manual definition of both modules combination and divided assertions, which reduces the automation of model-checking. Their success depends largely on the choice of abstract variables.

On the theoretical research of combination model-checking, the works use linear temporal logic (LTL) or introduce a temporal operator to TLA, which construct different combinatorial verifying frameworks and corresponding combinatorial verifying rules. Some methods can explain some cyclic verification rules in LTL and the cycle combination rules based on Moore machine model or Lattice theory [13].

On theorem proving for PLC program, some researches design simple PLC modular model and assume the current value increases monotonically and no resetting action course. Then the model is verified by theorem proving tool Isabelle/HOL. Others define PLC instruction semantics and verify a property on safety or time sequence by theorem proving tool COQ [14].

The mentioned work is so limited that they cannot be taken directly to verify general PLC programs. Moreover, they didn't implement the derivation of overall properties for whole system from sub-system properties.

III. The Framework for Compositional Verification of PLC Program

The above-mentioned studies cannot systematically solve the verification problems of PLC program. We combine the characteristics of the PLC program with the advantages of different verification means, and make PLC program verification technology and formal verification methods practically used.

PLC program is a type of embedded software and has its own characteristics:

- Most PLC programs are executed in embedded hardware environment. Their logical structure is relatively simple. They have short and less kind of statements.
- A PLC program is written by hardware instructions (or represented by ladder diagrams). So the abstract process of its model is relatively simple.
- PLC program still has most mechanisms of a high-level programming language. What the verification problems of PLC program need to deal with are similar as traditional program.
- PLC program is executed in sequence within one scanning cycle, and then next scanning cycle after refreshed output mapping. Inside a scanning cycle, it is similar to sequential programs. In whole scanning period, PLC shows as output responses for different input signals. Because cumulative values of various timers or counters are cross a scanning cycle, a PLC program cannot simply think of as logic responses transformed from input to output.

Compared PLC program with high-level language programs, its verification is more close to mathematical representation. Therefore, the verification of PLC program has its advantages.

The correctness of a PLC program should have the correctness of dynamic behaviors and static properties from macroscopic or external features. It should also have the correctness of coding and control timing sequence from microcosmic or internal features. The control timing sequence can be divided into that cross scanning cycle and within

scanning cycle. The two sequences influence the dynamic behaviors and the static properties of a PLC program.

The research combines the verification technologies on three levels to verify the correctness of PLC program, i.e. for PLC program coding correctness are verified on code level, dynamic behaviors (or cross scanning cycle) on model level and static properties (or within scanning cycle) on statute level. The hierarchy framework for compositional verification of PLC Program is showed as Fig.1.

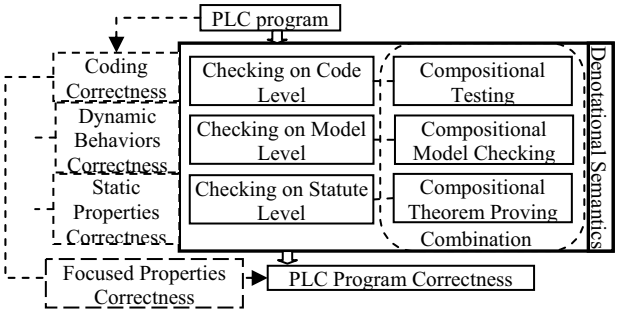

Fig.1 Hierarchy Framework for Compositional Verification of PLC Program

The framework is constructed by mutual complemented technologies including testing, model checking and theorem proving. The research needs easy to be used and realize automatic verification for PLC program.

IV. The Architecture of the Compositional Verification for PLC program

The flow and the constructed key elements of the compositional verification architecture are showed as Fig.2.

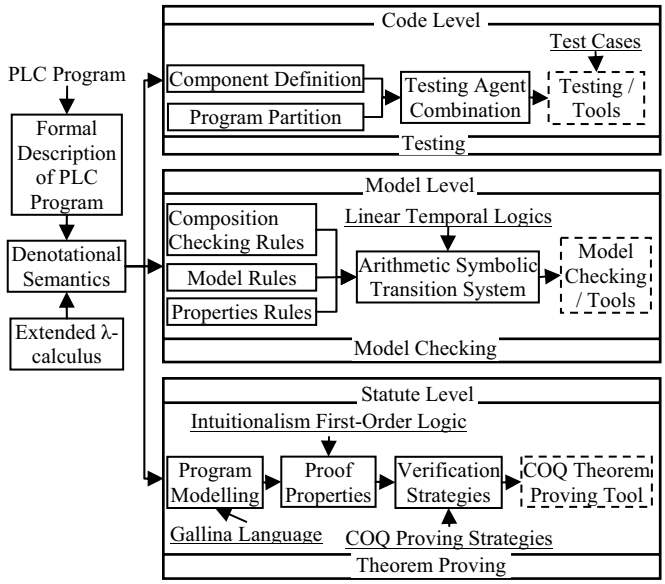

Fig.2 The Architecture of the Compositional Verification for PLC Program

Generally the obtained PLC program is the code. In order to obtain its model and high-level statute, conversion method must be provided from codes on low level to models on high level and statute on top level. Firstly, the formal description of PLC program and the structure of denotational semantics need to study. Because of the variables of register, bit and a series of stack operations, the extended definition of λ-calculus is studied and used to define PLC program.

Correctness on code level is required by the code correctness. PLC program is decomposed into different modules on basis of control objects and responses. The compositional testing solves the problem which testing cannot be executed in the real environment. It can reduce testing scale and improve testing coverage. Selected typical test cases are directly performed to find most of the bugs. Because testing method is very much dependent on test cases' selection, there will be fail-to-report bugs. To compensate this deficiency, PLC program verification is studied on model and statute level.

Correctness on model level is the correctness constraints of software design. From the perspective of software design, most key verification is control timing sequence cross scanning cycle in PLC program. It verifies the correctness of dynamic behaviors while PLC program runs. The design defects and mistakes are found on model level. Because of the existence of register variables in PLC program, the actual state space may be very large on model checking. The state space size is reduced by using combinatorial model checking to avoid "state space explosion" during verification. Model checking is also used to compensate the fail-to-report problems which are caused by insufficient testing coverage.

Correctness on statute level is the correctness constraints of PLC program requirements. Its constraint ensures the correctness of design and code for PLC program. The correctness of PLC software within a scanning cycle is verified by theorem proving. Model checking can find and report wrong routes by means of segmented or merged scan cycles. The theorem proving verification tests out whether the report is a false report. COQ is a main tool in the field of theorem proving. It is based on the calculus of inductive constructions, and has the powerful base of mathematical model and good expansibility. It has a complete set of tools, a full-time R & D team and supports open source. So the tool is chosen for theorem proving.

Three levels' technologies verify the correctness of functions and properties of PLC program. From different levels they ensure the correctness of PLC program on verified properties concerned by users.

V. Main Work and Achivements

The work focuses on testing and verification for PLC program. It includes the following aspects.

In the foundation, the work has proposed the verification architecture and methods of PLC program correctness. In order to ensure that combinatorial testing, model checking and theorem proving have the identical verifying semantics in the architecture, it has abstracted and described the typical PLC working modes, system models, the syntax and the semantics of PLC program. The mathematical description, configuration definition, several operations and related function definitions have been studied for the architecture, which are based on the above framework and the partitioned structure of PLC program. The formal description, the denotational semantics and

functions of PLC program are defined and constructed by extended λ-calculus used as a tool. The verification technologies on the three levels complement each other with identical basis.

On code level, corresponding to PLC software testing, the work analyzes the applicability of static testing, testing in real environment, hardware checker testing, instrumentation test and simulation test. It presents a combinatorial mechanism of software testing framework and testing method. Based on software components replacing function parts, high-level language code segments and compositional test modules are defined by equivalent denotational semantics and compositional testing strategy. The configuration is described and simulated by software. PLC program testing is converted to the equivalent testing of high level language program segments. The framework and method can make PLC software testing decomposed into several small testing. It solves the testing problems of restoring the real running environment, limit boundary and non-environment support. It also improves the testing coverage.

On model level, according to the PLC program architecture, formal description and semantics definition, arithmetic symbolic transition system is introduced by means of linear temporal logic syntax and semantics. It solves the problem that basic symbolic transition systems cannot accurately describe the system in practical application. To verify temporal properties of PLC circuit across scanning cycle, variables sets, predicates and migration functions are designed as transformation from a PLC program to a symbolic transition system. Some strategies of model checking are provided for the transition system. A set of the model and the properties of the composite verification rules are defined. They verify the correctness of dynamic behavior when PLC program runs, and reduce the verification scale. The defined compositional verification rules ensure the accuracy of verification strategies based on mathematical proof.

On statute level, the work presents a correctness verification framework based on theorem proving technology for PLC program. The correctness or static properties are verified in one scanning cycle. Based on COQ Gallina language, PLC program is modelled by the inductive conversion of the semantics structure. The denotational semantics is described to prove program properties. The work can deal with the model verification under infinite state space.

On application, a pendulum control system is chosen as a typical example and an experiment. PLC output drive module of the pendulum control system is verified by compositional verification method. It is tested under the boundary limit conditions and non-environment support. Its correctness properties such as system safety, liveness and fairness are verified by model checking and theorem proof under compositional strategies. Experimental results are compared with the general condition and show that the framework has effectiveness and advantage.

VI. CONCLUSION

The paper briefly introduces the compositional verification framework for PLC programs. The specific strategies, rules, function and model in the framework and architecture can be referred to Reference [15]-[17]. Although the research on the framework and architecture has obtained some achievements, some works need further to study. For example, model checking and theorem proving tools need to develop. The tools should support that PLC programming language such as IL and ladder diagram is automatically converted to models and verification function. They will enhance the verification usability.

ACKNOWLEDGMENT

The authors would like to thank all colleagues who contribute to this study.

REFERENCES

[1] Lewis R. Programming industrial control systems using IEC 1131-3, volume 50 of Control Engineering Series. Stevenage, United Kingdom: The Institution of Electrical Engineers, 1998.

[2] B. Kang, et al. A Design and Test Technique for Embedded Software. Proceedings of the 2005 Third ACIS Int'l Conference on Software Engineering Research, Management and Applications. Michigan: Mount Pleasant, 2005: 160-165.

[3] Mertke T, Frey G. Formal Verification of PLC-programs generated from Signal Interpreted Petri Nets. Proceedings of Proceedings of the SMC 2001, Tucson (AZ) USA, 2001: 2700-2705.

[4] H. S. Hong, et al. Data flow testing as model checking, The 25th International Conference on Software Engineering. IEEE Computer Society, US: Portland, 2003:232-242.

[5] The Spin homepage: http://spinroot.com/spin/whatispin.html.

[6] The NuSMV homepage: http://nusmv.irst.itc.it/.

[7] The UPPAAL homepage: http://www.uppaal.com/.

[8] S. Owre, S. P. Rajan, J. M. Rushby, N. Shankar, M. K. Srivas. PVS: combining specifications, proof checking and model checking. R. Alur and T. A. Henzinger, eds. LNCS: CAV'96, 1996, 1102: 411-414.

[9] The COQ toolkit. http://COQ.inria.fr/.

[10] R. S. Boyer, J. S. Moore. Proving theorems about lisp functions. Journal of the ACM, 1975, 22(1): 129-144.

[11] V. Gourcuff, O. de Smet, J. M. Faure. Improving large-sized PLC programs verification using abstractions. Proceedings of the 17th World Congress on The International Federation of Automatic Control, Seoul, Korea, July, 2008: 5101-5106.

[12] Bastian Schlich, Jorg Brauer, Jorg Wernerus, Stefan Kowalewski. Direct Model-checking of PLC Programs in IL. Proceedings of 2nd IFAC Workshop on Dependable Control of Discrete Systems. École Normale Supérieure de Cachan, Italy, 2009: Vol2(1).

[13] P.Maier. A Lattice-Theoretic Framework For Circular Assume-Guarantee Reasoning [PhD thesis]. Saarbryucken: University at des Saarlandes, 2003.

[14] Jan Olaf Blech, Sidi Ould Biha. Verification of PLC properties based on formal semantics in Coq. Proceedings of the 9th international conference on Software engineering and formal methods (SEFM'11). Springer-Verlag Berlin, Heidelberg, 2011: 58-73.

[15] Xiao Litian, Gu M, Sun Jiaguang. The Denotational Semantics Definition of PLC Programs Based on Extended λ -Calculus. Communications in Computer and Information Science, 2011, 176(II): 40-46.

[16] Litian Xiao, Rui Wang, Ming Gu, Jiaguang Sun. Semantic characterization of programmable logic controller programs. Mathematical and Computer Modelling, 2012, 55(5-6): 1819-1824.

[17] Xiao Litian, Gu Ming, Sun Jiaguang. The Verification of PLC Program Based on Interactive Theorem Proving Tool COQ. Proceedings of 4th IEEE International Conference on Computer Science and Information Technology(ICCSIT2011), Chengdu, China, pp.374-378, June, 2011.