A Model Projection Technique for Compositional Verification using Model Checking

Dong-Ah Lee[O], Junbeom Yoo
Department of Computer Science and Engineering, Konkuk University
{ldalove, jbyoo}@konkuk.ac.kr

모델 체킹을 활용한 구성 검증을 위한 모델 투영 기법

이동아[O], 유준범
건국대학교, 컴퓨터공학과

Summary

Model checking of large-scale software is hardly possible because the software is too complex and large. Model checking of a component which is a small part of the software is acceptable to avoid the state explosion problem where the component is not huge and too complex. This paper introduces a novel technique, called model projection, for compositional verification using model checking. The model projection is an activity to identify proper parts of a whole system with respect to verification purposes in order to apply model checking to the large-scale software.

1. Introduction

Model checking is one of formal verification techniques, which checks that a model meets its specification with the aid of automated support [1]. The technique is possible to model a system at any level of the system hierarchically, and to make properties along the level of the system hierarchy. The checking, however, restricts the model not only to have a finite number of states but also to have the size and complexity which a checking algorithm is able to search state space of the model exhaustively. The state explosion problem [2] may arise, as a model is too huge and complex.

Verification at a component level using model checking is acceptable to avoid the state explosion problem where a single component is not huge and complex. Model checking of a component checks that the component meets local properties at the same level of detail. Just because results of the checking show that a model of the component satisfies the properties does not mean that the component correctly works in the global system. It depends on how the component affects the whole system. Furthermore, model checking compositionally requires analysis of relations and influence between components at a system level in order to assert that the results are valid at the system level.

This paper introduces a novel technique, called *model projection*, for compositional verification using model checking techniques. The model projection is an activity to identify proper parts of a whole system with respect to verification purposes. The technique projects small models (formal models at a component-level) from a large one (a formal model at a system-level) in order to verify the proper parts using model checking. The model projection reveals relations and influences between the identified parts and the whole system.

2. The model projection technique

The model projection is a technical process to identify relevant parts simulating a formal model at a system-level with scenarios derived from a verification requirement. <Fig. 1> describes a brief process of the technique. Modeling FM_s (formal model at a system-level) refers software requirement or design specifications and generating S_i (i_{th} simulation scenario) refers verification requirements. Because of the size and complexity, the FM_s is usually much abstracted, which does not have specific information. Simulation of FM_s with S_i identifies P_i (parts projected by the i_{th} simulation scenario) which are running parts of the system with respect to the S_i.

The P_i is a target for the compositional verification using model checking. Traceability analysis assists identifying specific information for modeling with specific information. The analysis traces the P_i to relevant parts of source codes, C_i. Finally, we have a formal model, fm_i,

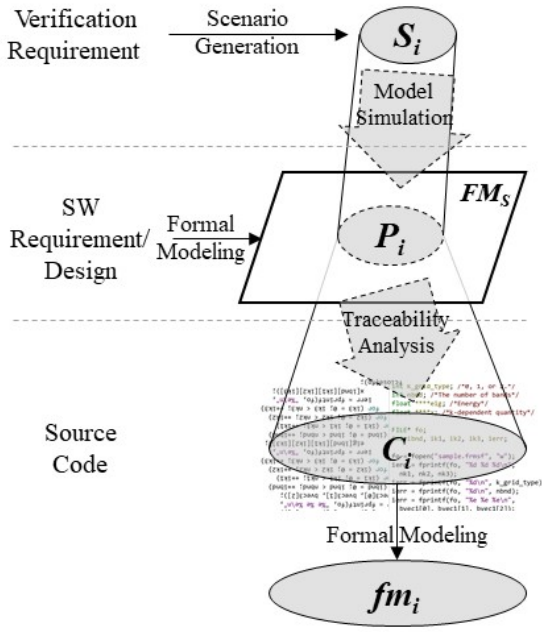

Figure 1. The model projection by simulation

which includes running parts by simulation of FM_s with S_i. Verification properties for model checking of fm_i can be both local properties of P_i and global properties of S_i. The model checking of fm_i is another phase of the compositional verification, which requires effort on selection of a model checker, property specification, analysis of its results and so on. Because the model checking itself is not a main idea in the paper, we omitted the process in detail.

The one of easiest ways to project the parts (P_i) is to model the FM_s with a tool which can model and simulate a system at the system-level, for instance, Statemate [3]. Statemate is a reasonable tool for modeling and verifying large and complex system at a system-level, since it is possible to model hierarchies of the system using various types of graphical languages, such as a Module-chart, an Activity-chart, a Flow-chart, a State-chart, etc. Statemate indicates active parts (Pi) through coloring the active parts during a simulation. The indicating active parts are a starting point of traceability analysis to identify relevant parts in source codes (C_i) with a simulation scenario (S_i).

3. Case study

Qplus-AIR [4] is a real-time operating system complying the ARINC 653 specification [5], which ETRI developed for avionics. Formal modeling and simulation of Qplus-AIR uses Statemate and the modeling refers a software design specification (SDS) of Qplus-AIR. <Fig. 2> shows an overall model of Qplus-AIR at a system-level of the operating system.

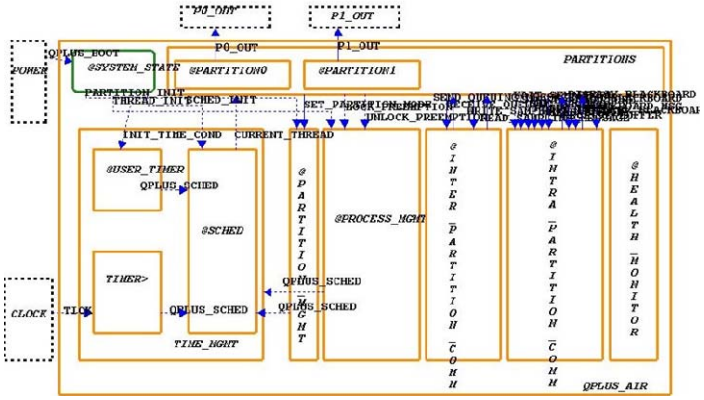

Figure 2. The formal model of a system level of Qplus-AIR with Statemate

Generation of simulation scenarios refers ARINC 653 specification, as the Qplus-AIR should comply requirements in the specification. Simulation and compositional verification in this paper focus on transitions of partition modes—the partition is one of key services in ARINC 653 to execute one or more avionics applications independently. There are 4 operating modes and transitions as described in <Fig. 3>.

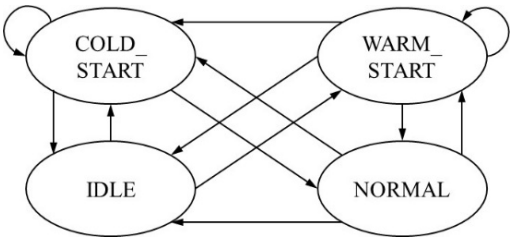

Figure 3. Partition modes and transitions in ARINC 653

Simulation scenarios only include the transitions between modes. Results of the simulation indicated small parts of the model color activated charts in violet. We found 9 functions and one major variable are related with the simulation scenarios through traceability analysis with the SDS, the formal model with Statemate, and source codes.

Model checking uses CBMC [6], which is a bounded model checker for programs written in C or C++ programming language. CBMC is available for Qplus-AIR because it is written in the C. We put an assumption statement and an assertion statement in source code to check the possibility that the mode transition from IDLE to NORMAL. The assumption means pre-condition of a partition's mode, which is IDLE in the case. The assertion

specifies post-condition of the mode and a return value for the transition function, NORMAL and no error. The assertion statement in the source code below:

```
assert((partition_mode == NORMAL)
       && (return == QPLUS_ERRNO_OK));
```

If the conditions in the assertion evaluate to true, then the mode transition from IDLE to NORMAL is possible. <Fig. 4> shows a screen dump of the verification result whether the conditions evaluate to true or false. “VERIFICATION SUCCESSFUL (no false evaluation)” at the bottom of the result means that IDLE to NORMAL without errors is possible in source code of Qplus-AIR.

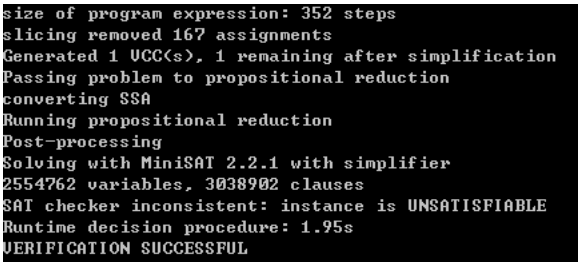

Figure 4. The verification result of a mode transition from IDLE to NORMAL by CBMC

Not only software development documents of Qplus-AIR but also ARINC 653 does not specify anything about the result. ARINC 653 does not give a result of the transition, while other invalid requests, such as a transition from COLD_START to WARM_START, are in requirements. Nevertheless, scheduling a partition without initialization is not normal behavior, and undefined mode transitions should not be allowed even if they are not possible to occur in current situation. It must be only possible within a strong assumption that the operating system, applications, or any kind of components in aircraft do not request the API, SET_PARTITION_MODE, to change a partition’s mode to NORMAL when it is IDLE.

4. Conclusions

The paper introduced the model projection technique to identify relevant parts with verification purposes. Compositional verification without systematic analysis about relation and influence between components at a system level is easy to lose confidence that results of the verification are still valid at the system level. The model projection identifies the relations and influence through simulation of a formal model of a whole system and reveals the relevant parts.

We are now developing an elaborate model and a tool for traceability analysis to make projection much easier and quicker. Since the traceability in hand requires a large amount of effort on finding traces between documents, models, and source codes. If all information is modeled properly, tracing between the information requires less effort. Furthermore, an assistant tool we plan may let the traceability analysis be semi-automatic or full-automatic.

Acknowledgement

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation (NRF) of Korea funded by the Ministry of Science, ICT (NRF-2017M3C4A7066479) and Basic Science Research Program through NRF funded by the Ministry of Education (NRF-2017R1D1A1B0303 0065).

References

[1] E. Clarke, *et al.*, Model Checking, MIT Press, 2018.
[2] E. Clarke, *et al.*, Progress on the state explosion problem in model checking., Informatics, pp. 176–194, 2001.
[3] David Harel, *et al.*, Statemate: A working environment for the development of complex reactive systems, IEEE Transactions on software engineering, Vol. 16, No. 4, pp. 403-414, 1990.
[4] T. Kim, *et al.*, Qplus/Esto-AIR: DO-178B Level A Certified RTOS and IDE for Supporting ARINC 653, Communications of the Korean Institute of Information Scientists and Engineers, Vol. 30 No. 9, 65-70, 2012.
[5] Airlines Electronic Engineering Committee, Avionics Application Software Standard Interface ARINC Specification 653 Part 1, Aeronautical Radio Inc., 2006.
[6] E. Clarke, D. Kroening, and F. Lerda, “A Tool for Checking ANSI-C Programs,” Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004), LNCS 2988, 168-176, 2004.