On Using Abstraction to Model Check Distributed Diagnostic Protocols

Péter Bokor
Advisor: András Pataricza

I. Introduction

In my master thesis I presented a method to verify a specific diagnostic algorithm (called DD [1]) by using formal methods. I applied model checking as the means of verification, which performs exhaustive simulation of the system model. Even for few diagnostic nodes (processing elements, also called PEs) the simulation yielded state space explosion. In order not to lose on generality of the verification we introduced an abstraction of the system. The novelty of the approach was to exploit the symmetry of the problem which was arisen from the fact that non-faulty nodes (PEs) show similar behaviour. In fact, by applying the *one-correct-node abstraction*, i.e. despite modeling each correct node separately we defined one abstract node representing the correct ones, the size of the state space became feasible. To prove the soundness of the abstraction we pointed out that by non-deterministic assignment of the node variables all scenarios of the real system can be modeled.

Our ongoing research is aimed at verifying another diagnostic protocol. In our former work, even for the abstracted system, we restricted ourselves to a certain number of nodes assuming that the requirements (see in Section II.B.) also hold for an arbitrary number of PEs. The new diagnostic protocol also makes more realistic assumptions on the communication network (duplicated bus instead of fully connected network) and exposes further challenging issues (e.g. maintenance-oriented diagnosis).

II. The Diagnosis Problem

A. Application Field

The diagnostic protocol under verification has been developed in the DECOS project. DECOS [2] (Dependable Embedded Components and Systems) is an EU project aimed at establishing a framework for integrated design of dependable (embedded) systems. Instead of dealing with the *federated approach*, where each service of the system is implemented by a dedicated component, DECOS distributes the services (called DASs, Distributed Application Subsystems in the DECOS terminology) on the installed nodes (i.e. processing units equipped by local memory, network connector device, sensors, actuators, etc.). The *integrated architecture*, in contrast to the federated approach, has many advantages, e.g. it facilitates the application of COTS (Commercial-Off-The-Shelf) components in dependable systems and also enhances the cost-effectiveness of the design process.

B. Requirements

Generally, if verifying a diagnostic protocol we may want to show that the protocol is *complete* and *correct*. While correctness requires that non-faulty nodes will never be accused to be faulty (safety property), completeness assures that every faulty node will be detected eventually (liveness property). In many cases correctness and completeness are contradictory requirements that need a tradeoff to be satisfied. In fact, in case of a non-restrictive fault model ensuring correctness implies that completeness only holds under certain restrictions to the fault manifestation.

C. The Diagnostic Protocol

If dealing with verification of diagnostic algorithms the requirements completeness and correctness are general properties. However, under which constraints they are to be guaranteed depends greatly on the system model (communication, fault hypothesis, etc.). While many approaches assume a fully connected network (like DD also does), which is not applicable in case of bigger systems, DECOS applies a Time Triggered Architecture (TTA). In TTA each node is connected to a duplicated bus and a scheduler a priori (at design time) assigns time slots to the nodes, where special devices (bus guardians) assure that in each slot only one node (the a priori scheduled one) issues messages to the bus. A detailed introduction of the DECOS diagnosis would be out of the scope of this paper, thus we only give the basic assumptions: (1) no restrictions on communication faults, (2) host faults are symmetric and (3) the diagnostic service may be either fail-silent or Byzantine (design decision).

Count-and-threshold [3] mechanisms are widely used in many application areas. The main idea is that each time a fault effect is recorded, a penalty value to the corresponding DUT (device under test; in DECOS the DUT is the node) will be assigned to indicate an increased evidence of the presence of the fault. When a fault-free behaviour is recorded after the detection of some fault effect, the penalty is progressively decreased, denoting a major trust on the absence of a permanent fault. In the DECOS diagnosis count-and-threshold will be a core part of the protocol (unlike in DD), hence its consideration in the verification process is of crucial importance.

III. Verification

A. *Generality of the Verification*

While verifying DD the straightforward encoding of the algorithm turned out to be infeasible for model checking (even for 4 nodes). With the one-correct-node abstraction we managed to reduce the state space, however, the model remained bounded to a given number of nodes: the model variables encoding the number of PEs could not be defined unbounded for finite model checkers. Now we are aimed at proving the protocol without any loss on generality.

B. *Verification Environment and Techniques*

As a verification platform we are using the SAL framework [4], which provides a ”family” of model checkers. Beside the standard approach (symbolic model checking) it also provides a *bounded model checker*. BMC (Bounded Model Checking) [5] may increase the efficiency of model checking and also supports *infinite model checking*. However, the approach is only applicable for proving invariants (note that completeness is an eventual property), and also the implementation in SAL is an open issue. More research on that is planned to be done in the upcoming months.

References

[1] C. Walter., P. Lincoln, and N. Suri, “Formally verified on-line diagnosis,” *IEEE Transactions on Software Engineering*, Nov. 1997.

[2] H. Kopetz, R. Obermaisser, P. Peti, and N. Suri, “From a federated to an integrated architecture for dependable real-time embedded systems,” Tech. Rep.

[3] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, “Threshold-based mechanisms to discriminate transient from intermittent faults,” Technical Report B4-17-06-98, IEI-CNR, June 17 1998.

[4] S. Bensalem, V. Ganesh, Y. Lakhnech, C. Muñoz, S. Owre, H. Rueß, J. Rushby, V. Rusu, H. Saïdi, N. Shankar, E. Singerman, and A. Tiwari, “An overview of SAL,” in *LFM 2000: Fifth NASA Langley Formal Methods Workshop*, C. M. Holloway, Ed., pp. 187–196, Hampton, VA, June 2000. NASA Langley Research Center.

[5] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, “Symbolic model checking without BDDs,” in *TACAS ’99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pp. 193–207. Springer-Verlag, 1999.