Semi-decisions in the validation of dependable systems [1]

András Pataricza
Budapest University of Technology and Economics, Dept. Measurement and Information Systems
pataric@mit.bme.hu

1. Introduction

The growing complexity of information systems necessitates a complete automation in proving the conformance to the functional and dependability related requirements. Traditional quality assurance methods are unable to provide a thoroughgoing check any more. This way, the fulfillment of the specification has to be proved by mathematical modeling based evaluation.

However, even advanced analysis methods are unable currently to cope with faithful models, due to the computational complexity of verification problems. In large-scale models the size of the manageable state spaces confines to the order of magnitude of 10^{120}. Moreover, dependability analysis frequently leads to the necessity of exploring the entire state space, for instance, to prove that the system will never reach an unsafe state.

An alternate solution is to examine the violation of the objective requirements in a larger, but easier to generate state space embedding the state space of the system under evaluation as a subspace. If no counterexample is found in the embedding space, then obviously the safety property is fullfilled in any subspace of it, including the state space of the target design, as well.

However, if a counterexample is found, it cannot be decided, whether it belongs to the state space of the target design (i.e. it is a true counterexample) or only a spurious solution from the extension of the state space. This way, a semi-decision algorithm can deliver the alternatives “Yes” or “I don’t know”.

The current paper reports about an ongoing research to prove the correctness of designs described in UML. The most specific feature of the approach taken is the objective to prove the correct functioning of the target design even in the presence of faults anticipated in a predefined fault model [4].

2. Transformation-based validation and verification

One of the most accute challenges in the design of information technology devices is the verification of complex systems. Visual design technologies, like UML, the Unified Modelling Language, increase the design quality and productivity, however, they are unable to completely exclude design errors.

Roughly speaking, these new CASE tools will play a similar role in the development of design technologies like the use of high-level languages instead of assembler level programming. The semi-formal specification of the target design in UML makes the application of mathematical proof of correctness technologies a favorite candidate [6,7]. Several research projects aim at an automatic transformation and back annotation based analysis, design validation and verification [5].

However, transformation-based analysis methods cannot perform the evaluation of larger models than the native approach directly building the model in the input language of the mathematical tool. Moreover, due to the redundancy in the automatically generated models the manageable model size is slightly reduced. This way the development of approximate methods that are able to handle large scale models as well, becomes a crucial factor in the dependability assurance process.

3. Basics of semi-decisions in Petri-Nets

One of the favorite description paradigms for automatic transformation based analysis is the Petri-nets, the traditional modeling paradigms for non-deterministic, concurrent systems. When using non-interpreted modeling [3], where data-dependencies and values are substituted with a non-deterministic abstraction, Petri-nets are able to model extremely large systems as well.

As already mentioned in the introduction, semi-decisions necessitate a proper covering space for embedding the solution space. In the case of Petri-nets, such an abstraction is provided by the state equation [2]. This state equation summarizes the difference of the numbers of tokens at the individual places of a Petri-net between the initial and final markings of a firing sequence. The state equation corresponds to a temporal compaction of the control flow in the modeled system, as it contains only the numbers of occurrences of the individual elementary

[1] This work was supported by the OTKA-T030804 and FKP 0193/1999.

state transitions, but it does not define their execution order. Moreover, if a sequence satisfies the state equation, this is only a necessary, but not always a satisfactory condition of its execution.

The state equation is a proper basis for semi-decisions, as it provides the structural specification of the target system under evaluation in a compact form. The generation of the state equation, (a linear Diophantine inhomogeneous equation system can be reduced to a number of basis vectors.

Problem specific temporal constrains can be formulated either directly in a linear algebraic [1] or in the form of a temporal logic expression from which a similar solution methodology can be applied [8]. If the state equation is handled over the real numbers instead of the natural ones (real relaxation), state spaces up to 10^{500} can be analyzed as well [9].

4. Applications of semi-decisions in dependability evaluation

A number of questions related to dependability requires the simultaneous analysis of all possible faulty instances of the target system in addition to the fault-free one. Typical criteria for safety related medical equipment require for instance the tolerance of an active and a latent error. Similarly, an important question is the number of faults tolerated by a system or the probability of a failure.

Semi-decisions can be used to answer to this category of questions in the following way:

- At first, a combined model is constructed which incorporates both the fault-free and all anticipated faults into a single, integrated model.
- If the number of active faults is constrained, for instance by the single fault assumption, this can be added to the model either in a structural way (by introducing selectors activating individual faults) or in an algebraic form (by adding an equation limiting the number of fault activation).

Depending on the objective of the analysis, different algebraic analysis methods can be used:

- If a negative proof is aimed at (for instance it has to be proven that no unsafe state will be reached), then the unfeasibility of the state equation and constrains has to be proven.
- If a numerical attribute has to be proven (for instance the number of faults belonging to some categories has to reach a predefined limit), a proper objective function expressing the number of faults leads to an integer programming problem.
- Similarly, searches for extremal values of probabilistic measures can be solved by a MILP [10].

5. Pilot applications

In the framework of an ongoing research the technologies sketched above are applied to the safety validation and verification of an artificial kidney controller, a wireless interconnection protocol and a railway control system for which efficiency measures will be collected.

12. References

[1] J. Desel: Petrinetze, lineare Algebra und lineare Programmierung, Teubner-Texte zur Informatik, 26., B.G. Teubner, Stuttgart-Leipzig, 1998
[2] M. Silva, E. Teurel, J.M. Colom: Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems in W. Reisig, G. Rozenberg (Eds.): Lectures on Petri Nets I: Basic Models, Springer LNCS 1491, 1998
[3] Gy. Csertán, A. Pataricza, E. Selényi: Dependability analysis in HW-SW co-design. In IEEE Computer Performance and Dependability Symposium, IPDS'95, pp 306-315, 1995.
[4] A. Pataricza: Algebraic modeling of diagnostic problems in HW-SW co-design. In D. Avresky, B. Horst editors, Digest of Abstracts of the IEEE International Workshop on Embedded Fault-Tolerant Systems, Dallas, Texas, 1996.
[5] A. Bondavalli, M. Dal Cin, D. Latella, A. Pataricza: High-level Integrated Design Environment for Dependability (HIDE), Proceedings of Words'99F: IEEE CS Fifth International Workshop on Object-oriented Real-time Dependable Systems, 1999.
[6] D. Varró, G. Varró, A. Pataricza: Designing the Automatic Transformation of Visual Languages. Accepted for *the Journal of Science of Computer Programming.*
[7] D. Varró, Sz. Gyapay, A. Pataricza: Automatic Transformation of UML Models for System Verification. In J. Whittle et al. (eds) WTUML'01: Workshop on Transformations in UML, Genova, 2001
[8] J. Esparza and S. Melzer. Verification of safety properties using integer programming: Beyond the state equation. *Formal Methods in System Design*, 16:159-189, 2000.
[9] S.Dellacherie, S. Devulder, J.-L. Lambert: Software Verification Based on Linear Programming, In: J.M. Wing, J. Woodcock, J. Davies (Eds.): FM'99 - Formal Methods World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, Proceedings, Vol. II, Springer LNCS 1709, 1999
[10] B. Polgár, Sz. Nováki, A. Pataricza, F. Friedler: A Process-Graph Based Formulation of the Syndrome Decoding Problem. IEEE DDECS, Design and Diagnostics of Electronic Circuits and Systems, pp. 267-272