Domain-specific model checking: exhaustive testing in model-based development (Position paper)

Mika Cohen

FOI - Swedish Defence Research Agency

***Abstract*—Model-based development in the defence industry opens up the door for exhaustive testing of safety critical systems by means of model checking.**

I. Domain-specific modeling

Domain-specific modeling (DSM) is rapidly gaining acceptance within the defense industry and related industries such as the avionics and automotive industries. Studies show that DSM is consistently 5 to 10 times more productive than manual coding practices and reduces the number of code errors by 50% (cf. [1] , [2]).

In DSM, the developer specifies the solution using familiar concepts taken directly from the problem domain; a mobile phone app, e.g., might be specified in terms of *query*, *pop-up*, *SMS*, etc. The developer thus focuses on specifying the intended functionality rather than tedious implementation details. The final product, i.e., complete source code, is generated automatically from the high-level designs. The automated code generation is possible precisely because the modeling language is specific to a narrow problem domain.

The narrow problem also enables more effective program verification: designs can be checked for consistency with respect to domain specific rules, such as a rule stating that a start state may have at most one outgoing transition. In this way, the modeling language can prevent modelers from creating designs that are illegal or lead to errors.

In fact, DSM even opens up the door in industry for software model checking, the holy grail in program verification.

II. Model checking

Model checkers are tools that automatically verify that a system cannot exhibit some specified undesirable behaviour such as deadlock, livelock, information leakage, etc. The model checker explores all logically possible execution traces of the system, making the verification equivalent to exhaustive testing. The model checker returns an execution trace exhibiting the unintended behaviour (e.g., deadlock) if such a trace exists, a useful feature during debugging.

Model checking is extensively used in the hardware industry [3]. Until recently, however, model checking *software* has not met with much success in industry. The main obstacle has been the effort required to produce designs that can be analyzed by model checkers - model checkers take abstract models as input rather than concrete source code. Users had to manually construct an abstraction (of the source code) in the input language of the model checker, introducing additional cost and delay.

This situation has changed with the adoption of DSM in industry: the high-level designs produced in DSM can be automatically compiled to a format readable by a model checker. Thus, exhaustive testing by means of model checking becomes push-button in a DSM context.

III. Example: unmanned aerial vehicle

A recent case study sponsored by the US Air Force compared the effectiveness of model checking and testing in the verification of the operational flight program of an unmanned aerial vehicle developed by Lockheed Martin Aerospace [4]. Two verification teams were set up, one focusing on model checking MATLAB Simulink/Stateflow designs and the other on traditional testing. While the model checking team uncovered 12 errors (none affecting flight safety), the testing team failed to find any errors. Both teams concluded that model checking was more effective than testing in finding design errors.

IV. Looking ahead: domain-specific probabilistic model checking

Real-time and probabilities are essential to performance and safety analysis of safety critical systems and other DSM application areas. Looking ahead, it would be interesting therefore to consider feeding DSM designs to real-time model checkers or even probabilistic model checkers that take timed models or timed-probabilistic models as input, e.g., timed-automata or Markov decision processes. Both real-time and probabilistic model checking are currently hot topics in academia.

References

[1] S. Kelly and J. Tolvanen, "Domain-specific modeling", IEEE Computer Society, 2008.

[2] J.L. Hammond, "Domain-specific modeling significantly reduces development time", Embedded Control Europe (ECE), 2008.

[3] L. Fix, "Fifteen years of formal property verification in Intel", 25 years of model checking 2008.

[4] M. Whalen et al., "Integration of formal analysis into a model-based software development process", Formal Methods for Industrial Critical Systems (FMICS), 2008.