Simulink to NuSMV Model Translation Challenges

Ganesha
International Institute of Information Technology
Bangalore, India
ganesha@iiitb.ac.in

Abstract

Safety critical systems must be formally verified to prove their correctness in order to avoid the loss of human life or a huge investment. Simulink is widely used in the industry to model safety-critical systems. NuSMV is the model checker which can be used to verify the models. To verify the designed models, developers need to be familiar with formal verification tools, languages, and syntax. The automatic translation of created models into the model checker's input language aids in the reduction of development time and expense. Model checking performed during the design phase detects system flaws early in the software development process. When numerous model verification tools are available, it is beneficial to implement an intermediate representation of the translated model. This paper discusses challenges faced in translating Simulink-based models into input language of a model checker.

Keywords: Modeling, Simulation, NuSMV, Simulink, Translation

1 Modelling and Simulation

A model is a simplified representation of a real system, whereas simulation is a computerised representation of a model that depicts the model's behaviour. Modelling and simulation is extensively used as a debugging and a comprehension step in the design phase of any safety-critical system. Simulink is the modelling environment embedded in MATLAB and provides a graphical environment for representing safety-critical systems. The Simulink models are represented by a block array in the Simulink library. This block diagram-based environment also provides simulation features. When designing components for these safety-critical systems, it is important to ensure that the components are reliable. To verify the requirements of these systems, you may manually provide test entries and verify the results. However, as the number of system components increases, it becomes increasingly difficult to manually inspect. Constructing a mathematical model of the system and translating requirements in mathematical notations allows us to perform model checking. Several model checking tools are available for this purpose [5].

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2 State Space in Safety Critical Systems

Safety-critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment. The computer programs which are complex in nature and used to control safety critical systems such as medical devices, aircraft's, satellites, ATM machines, nuclear power plants, weapon systems (or any Embedded systems) etc need to be verified for their correctness. Because the system's misbehaviour could have a direct impact on human life and result in a significant loss of investment. A variety of tools and techniques are used to test safety critical systems. Testing is not exhaustive, and it does not cover all possible system behaviours.

Any system behaviour that is ignored may result in system failure. As a results, developing a reliable safety critical system is very critical by considering all possible states in it. Safety critical systems, in general, have a huge state space and are difficult to verify. This is due to the inclusion of several subsystems in the design, the large number of variables involved, and the hardware and software interconnections, among other factors. One example of a huge state space is the integer domain. If we ignore the system's memory constraints, the program with just integer data types may have an infinite number of states.

3 Developer Challenges

The bulk of system defects can be traced to requirement and design phases when compared to coding phase. Before going through the actual development process, safety-critical systems go through a model-based design and simulation procedure. Systems modelled using Simulink can be formally verified to prove their correctness. Model checking is one of the formal verification techniques that uses mathematical reasoning to prove the absence of errors. The Model Verifier (NuSMV) accepts the specification and model that the final system is expected to meet. The tool verifies the correctness of the model and generates an appropriate counter example if it detects a defect.

However, software developers find it difficult to understand and learn various formal verification languages, their syntax and tool notations. An alternative approach is for software engineers to build models using Simulink, and use an automated translator to translate the model into the input format of a model checker like NuSMV. When translating a model into a model verifier input language, it is very important to preserve the semantics of the model. Researchers have made an attempt to automatically translate the models into the input language of a model checker [2]. However, there is more space for research in model translation when we look at the full list of Simulink library blocks.

4 Model Translation Challenges

The model checker NuSMV is a finite state verification tool and is limited to discrete Simulink models only. Also representation of real numbers is not supported by the NuSMV and variable domains must be bounded in NuSMV.

This section outlines the translation challenges by referencing the Simulink and NuSMV tools. Since Simulink has a wide variety of library blocks, certain common features are recognized and clubbed together. They are, Continuous and Discrete blocks, Lookup Tables, Logic Bit Operations, Math Operation etc. Among these, Boolean logic gates and simple mathematical block translation are straight forward to translate. Consider the process of translating a combinatorial circuit that is simulated using Simulink.Our approach considers circuit wires as variables and the logic block as computational units. The Fig. 1. below illustrates that this interpretation. One of the challenges during translation process is posed by integer variables. The integer domain is infinite in theory. We know that the state of a program is an evaluation of all the variables in the program. To find the errors in the program one has to consider any state that arise in the program. Reasoning for all possible states across all possible executions of a program is challenging due to large or infinite data domains. The model checker NuSMV model is intended for discrete, finite-state transition models. Direct translation of infinite domain to NuSMV model is impossible. In the citation we pointed in this paper is not able to translate the infinite domain into a finite domain.

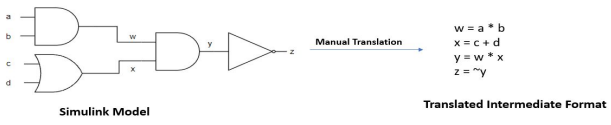

Figure 1. Simulink to NuSMV Translation.

In general there are many techniques or methods to solve state space explosion problem. Avoiding unnecessary state variables, Model slicing [1], removing duplicate clauses, abstraction [4], tightening variable domains, CEGAR [3] etc are a few among them. However we need to solve the problems in the domain of model translation as part of formally verifying safety critical systems. NuSMV does not support floating point data types. One such example is shown below in Fig. 2. which interacts between continuous and discrete blocks of a Simulink model. We also need to consider type inference as part of translation process to automatically deduce the data type of an evaluated expression.

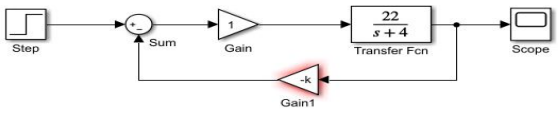

Figure 2. State models containing real numbers and logic circuit with feedback loops.

5 Conclusion and Future Research

In this paper we have described challenges observed during the translation of a model into the input language of a model checker. They are, a)A translation with an infinite domain (Say Integer Domain). b)Translating a model component that interacts with both continuous and discrete blocks.c)Translation of an expression which results in real numbers. d)Generalized translation algorithm in an intermediate representation to cover majority of model checking tools. e) Model checking tool limitations. Though we use Simulink as the modelling language, and NuSMV as model checking tool, one of the objective of this research is to design generalized translation schemes and intermediate representations that apply to a wide range of modelling environments and model checkers. The most pressing challenge is to solve the state space explosion problem using the techniques mentioned in the paper. Addressing the other concerns listed above will aid in the direction of future research in this field.

References

[1] KELLY ANDROUTSOPOULOS. 2013. *State-Based Model Slicing: A Survey.* ACM. Retrieved Nov 15, 2021 from https://discovery.ucl.ac.uk/id/eprint/1410594/1/csur13.pdf

[2] Sudeepa Roy Meenakshi Balasubramanian, Abhishek Bhatnagar. 2006. Automatic translation of Simulink models into the input language of a model checker. Publication of US20080086705A1. Retrieved Nov 15, 2021 from https://patents.google.com/patent/US7698668B2/en Patent No. US7698668B2, Filed 2006-10-10., Issued 2010-04-13.

[3] Cong Tian. 2014. *Making CEGAR More Efficient in Software Model Checking.* IEEE. Retrieved Nov 15, 2021 from https://ieeexplore.ieee.org/document/6895263

[4] Wikipedia. 2021. *Abstraction (computer science).* Wikipedia. Retrieved Dec 15, 2021 from https://en.wikipedia.org/wiki/Abstraction_(computer_science)

[5] wikipedia. 2021. *List of models checking tool.* Wikipedia. Retrieved Nov 15, 2021 from https://en.wikipedia.org/wiki/List_of_model_checking_tools