Abstract

Computer systems are so complex and crucial to our lives that we need to verify that they are correct and do not fail or risk facing enormous economical consequences, like in the case of the European Space Agency's Ariane 5 rocket, which self-destructed 37 seconds after launch because of a software malfunction, or loss of human lives, like the Therac-25 radiation therapy machine, which caused at least six deaths due to overdoses of radiation because the machine was not able to detect a human error. We would like to reduce the number of such errors or even prove their absence.

Many errors stem from incomplete and inconsistent specifications of the systems to construct, as they are often written in natural language text. We would instead like to create a formal specification. In order to do that, we create a formal model of the system we wish to construct, much like how an architect creates a blueprint of a house that is to be constructed.

A specification, in the form of a formal model, can then be verified using formal analysis methods. One such method is the reachability graph method, which basically explores all possible executions of the formal model by creating a graph where each node is a state of the formal model and each edge indicates that it is possible to go from the source to the destination state. Such a graph is called a reachability graph. The reachability graph method has the advantage that is can be implemented in a computer and made almost completely automatic. Unfortunately, the behaviour of a formal model can be very complex, so we will often need a reduction technique, which tries to explore only part of the behaviour or represent the behaviour more efficiently in the computer memory. This thesis presents two such reduction techniques. One reduction technique, the sweep-line method, uses a user-specified notion of progress to remove states that will never be encountered again from memory. This method has the disadvantage that the structure of the reachability graph is not preserved, so certain properties cannot be verified. In order to overcome that, we have extended the sweep-line method to also store the structure of the reachability graph in a memory-wise nearly-optimal manner. Another method, the ComBack method, avoids storing the states of the reachability graph altogether, by exploiting that any state can be reconstructed if we know how to get to it from the initial state. By storing a spanning tree of the reachability graph, rooted in the initial state, the ComBack method manages to represent the reachability graph very efficiently.

While a specification written as a formal model makes it possible to verify desired properties, it is often difficult or even impossible for domain experts, who know about the system we wish to construct, to validate that the formal model indeed corresponds to the desired system. In order to facilitate communication of the formal model, we create a visualisation of the behaviour of the formal model. The behaviour of the visualisation is completely defined by the formal model, and the visualisation makes it possible to provide input to the formal model. This thesis presents three papers on this topic. One presents

a tool, the BRITNeY Suite, which makes it possible to create visualisations of formal models. Another paper describes an industrial case study where formal models and visualisations have been used to create a prototype of a network protocol facilitating communication between computers moving from one wireless network to another. The third paper provides a formal game-theoretic framework for tying visualisations to formal models.

This thesis deals with making formal models look good and behave well. By creating a domain specific visualisation, we can make the model look good, and allow domain experts to understand them. By verifying the model using the reachability graph method, we can make the model behave well, by removing errors in the model, making the model better suited as specification. This thesis consists of two parts. Part I gives an overview of formal models, their analysis, and visualisation of them. Additionally, Part I describes the five papers, which are re-printed in Part II. Four of these papers have been published at conferences and one is submitted to a workshop.