Formal specification and model checking studies using Maude

2130012 Tsuyoshi Nakamura

In recent years, everything has become digitalized. Furthermore, technological innovation is accelerating, and new technologies such as AI and IoT are emerging, making systems even more important. In addition, software systems are becoming increasingly complex and large-scale, making it difficult to verify all cases through conventional review and testing. In such a situation, it is very important to ensure that systems operate safely as they evolve through continuous system updates. Model checking is one of the methods to achieve this. The purpose of this research project report is to enable the use of Maude to correctly handle model checking for invariance. For this purpose, we investigated the creation of formal specifications, description in Maude, and operation of model checking for invariance using several examples. We also used cases with intentionally injected errors to extract counterexamples, investigated the transitions leading up to the occurrence of counterexamples, and deepened our understandings of how the state explosion problem, one of the challenges in model checking, occurs and how to deal with it. The report also describes the issues (state explosion) and phenomena (phenomenon of infinite state creation) encountered in the course of the investigation and research, and summarizes them while applying countermeasures. We also discuss the challenges and considerations we felt as we proceeded, and summarize the challenges we will face in the near future.

This research project report consists of seven chapters. Chapter 1 describes the background and purpose of this project research and summarizes the structure of the following chapters. Chapter 2 defines the state machine and summarizes the set of reachable states of the state machine. There are a number of ways to represent states, but in this report, we represent a state as a soup of observable components by enclosing them in braces { and }. We describe the creation of formal specifications, description in Maude, and model checking for invariance using a mutual exclusion protocol based on test&set, which is an indivisible instruction. We also describe the behavior of invariant model checking using test&set with intentionally embedded errors. The transition process from the inclusion of errors to the extraction of counterexamples and the generation of counterexamples is also described. In Chapter 3, we describe the creation of formal specifications, description in Maude, and model checking for invariance using Qlock, a mutual exclusion protocol using indivisible queues. In this chapter, we describe the creation of formal specifications, description in Maude, and model checking for in-

variance using Qlock (FQlock), which does not satisfy mutual exclusion and intentionally embeds the following errors.

- Enqueuing a process identifier to the end of the shared queue is not done atomic
- Dequeuing the head (or top) element from the shared queue is not done atomic

Focusing on a single process, the report uses illustrations to explain how the state changes when a state transition is made. The structure of the infinite number of reachable states, which is one of the causes of state explosion in FQlock, is also described. Even if the structure of FQlock is such that an infinite number of reachable states are created, it is avoided by utilizing Maude's search command, which employs width-first search. We confirmed that even with Qlock that satisfies mutual exclusion, state explosion occurs when the number of processes is increased without completing model checking. We also surveyed the Divide & Conquer Approach, which is one of the existing studies on the state explosion problem. In Chapter 4, we describe the formal specification, description in Maude, and model checking using the Identity-Friend-or-Foe Authentication Protocol (IFF), which is a protocol for identifying friend and foe. In this chapter, we describe the creation of formal specifications, descriptions in Maude, and model checking for invariance using the following intentionally embedded adversary-identity-friend-or-foe (FIFF) protocols.

- Encryption without including the sender's identifier

This error causes the IFF (FIFF), which does not satisfy the discriminability of friend and foe, to be misidentified as a friend, even though it is originally an enemy. The state explosion problem was not avoided simply by reducing the number of variables used in Maude's code or by utilizing JAIST's Large Memory PC Cluster (LMPCC), which is a large memory computer. However, by explicitly classifying participants (or principals) of IFF into initiators (those who initiate IFF) and responders (those who receive the first message of IFF), it is now possible to deal with this problem on an individual PC without using the Large Memory PC Cluster (LMPCC), which is a large memory computer. In Chapter 5, based on the knowledge gained in the previous chapters, we describe the creation of formal specifications, description in Maude, and model checking for protocols using public keys and nonce using the Needham-Schroeder Public Key Authentication Protocol (NSPK). Here, we check whether the confidentiality of the nonce is

maintained. The transitions in which impersonation was established are described based on counterexamples. In Chapter 6, we describe the creation of formal specifications, description in Maude, and model checking using the Needham-Schroeder-Lowe Public Key Authentication Protocol (NSLPK), an improved version of NSPK, based on the countermeasures against the state explosion problem developed in the previous chapters. Model checking is then conducted. Here, the encryption pattern is used to simultaneously express who sent the message and to whom, and the number of message transmission functions and variables used in NSPK are reduced. After explicitly classifying participants (or principals) of NSPK into initiators (those who initiate NSPK) and responders (those who receive the first message of NSPK), the model checking experiments are conducted to conform whether the confidentiality of the nonce is maintained. Chapter 7 summarizes the findings of the project research. We took up the following two issues that we felt through the preparation of the research project report, and studied the outline of related existing research as well as the countermeasures.

- To counter the state explosion problem, it is necessary to reduce the number of reachable states. Reduction requires modification of the code, but it is dependent on the developer's understanding of the nature of the verification target and the level of modeling understanding.
- Although there are many proposals on the effectiveness and application of model checking, there are few cases where model checking is used in actual system development sites.

We should investigate liveness properties in addition to invariant properties, which were not covered in this report. Maude, which was used in this report, is capable of linear temporal logic (LTL) model checking and can also verify liveness properties. We also discuss what elements need to be studied when verifying the liveness properties, and survey the existing research.